

Package ‘prolfqua’

October 24, 2026

Type Package

Title Proteomics Label Free Quantification

Version 0.99.3

Description The prolfqua package integrates essential steps of the mass spectrometry based label free quantification data analysis workflow: quality control, data normalization, protein aggregation, modeling, hypothesis testing and sample size estimation. We use prolfqua to : develop highly customizable, visually appealing, and interactive data analysis reports in pdf or HTML format; to visualize and model simple experimental designs with a single explanatory variable and complex experiments with multiple factors; to benchmark data acquisition, data preprocessing or data modeling methods. We developed and improved the package by applying the ``Eating your own dog food" principle, when analysing dozens of proteomics datasets.

License MIT + file LICENSE

Encoding UTF-8

VignetteBuilder knitr

URL <https://github.com/fgcz/prolfqua>, <https://fgcz.github.io/prolfqua/>

BugReports <https://github.com/fgcz/prolfqua/issues>

Imports circlize, ComplexHeatmap, dplyr, forcats, ggplot2, ggbeeswarm, ggrepel, limma, lme4, lmerTest, logistf, MASS, methods, plotly, progress, purrr, rlang, stringi, tidyselect, tidyr, tibble, UpSetR, withr

Suggests BiocCheck, BiocStyle, corrplot, covr, crosstalk, cyclocomp, devtools, goodpractice, gridExtra, KernSmooth, knitr, limpa, lintr, pkgdown, preprocessCore, rcmdcheck, Rfit, rmarkdown, roxygen2, S4Vectors, SummarizedExperiment, testthat, vsn

RoxygenNote 7.3.3

Roxygen list(r6 = TRUE)

Depends R (>= 4.6)

biocViews MassSpectrometry, Proteomics, Metabolomics, Normalization, Preprocessing, QualityControl

git_url <https://git.bioconductor.org/packages/prolfqua>

git_branch devel
git_last_commit 0726bc3
git_last_commit_date 2026-10-23
Repository Bioconductor 3.24
Date/Publication 2026-10-23
Author Witold Wolski [aut] (ORCID: <<https://orcid.org/0000-0002-6468-120X>>),
 Functional Genomics Center Zurich [fnd],
 Christian Panse [cre] (ORCID: <<https://orcid.org/0000-0003-1975-3064>>)
Maintainer Christian Panse <cp@fgcz.ethz.ch>

Contents

prolfqua-package	6
.get_rob scales	7
.lfqdata_to_elist	7
.make_progress	8
.resolve_weights	8
adjust_p_values	9
AggregateLimpa	10
AggregateMedpolish	12
AggregateRlm	13
AggregateTopN	14
aggregate_intensity_top_n	15
AggregatorBase	16
AnalysisConfiguration	18
annotation_add_contrasts	21
AnovaExtractor	22
apply_to_response_matrix	23
build_contrast_analysis	24
build_model	26
build_model_glm_peptide	27
build_model_glm_protein	28
build_model_impute	29
build_model_limma	31
build_model_limma_impute	32
build_model_limma_voom	33
build_model_limma_voom_impute	35
build_model_limpa	36
build_model_logistf	38
center_to_reference_cfg	39
complete_cases	39
compute_borrowed_variance	40
compute_borrowed_variance_limma	41
compute_contrast	42
compute_lmer_contrast	43
compute_pooled	44
ContrastConfiguration	45
Contrasts	48
ContrastsBinomialNestedFacade	51
ContrastsDEqMSFacade	52

ContrastsDEqMSVoomFacade	54
ContrastsFacadeBase	55
ContrastsFirth	58
ContrastsFirthFacade	60
ContrastsFirthNestedFacade	61
ContrastsInterface	63
ContrastsLimma	66
ContrastsLimmaFacade	69
ContrastsLimmaImputeFacade	70
ContrastsLimmaVoomFacade	72
ContrastsLimmaVoomImputeFacade	74
ContrastsLimpaFacade	76
ContrastsLimpaNestedFacade	78
ContrastsLmerNestedFacade	80
ContrastsLMFacade	81
ContrastsLMImputeFacade	83
ContrastsModerated	85
ContrastsModeratedDEqMS	87
ContrastsPlotter	90
ContrastsRfitFacade	95
ContrastsRfitImputeFacade	97
ContrastsRLMFacade	99
ContrastsROPECA	100
ContrastsROPECANestedFacade	103
ContrastsTable	106
contrasts_linfc	108
data_checksummarizationrobust87	109
data_checksummarizerobust69	109
data_ionstar	110
df.residual.rfit_prolfqua	110
df.residual.rlm	111
encode_bin_resp	112
estimate_intensity	112
FACADE_REGISTRY	113
filter_proteins_by_peptide_count	114
find_d0_deqms	114
find_package_file	115
generate_contrasts	115
generate_contrasts_for_factor	116
get_anova_df	117
get_complete_model_fit	117
get_contrast	118
get_p_values_pbeta	119
get_rob scales	120
get_uniprot_id_from_fasta_header	121
group_label	121
HierarchyCountsSample	122
hierarchy_counts	124
impute_from_model	124
impute_refit_singular	126
interaction_contrasts	127
INTERNAL_FUNCTIONS_BY_FAMILY	128

is_contaminant	129
is_decoy	130
level_specific_contrasts	130
LFQData	131
LFQDataPlotter	137
LFQDataStats	142
LFQDataSummariser	146
LFQDataToSummarizedExperiment	149
LFQDataTransformer	150
lfq_power_t_test_proteins	153
lfq_power_t_test_quantiles_V2	154
linfct_from_model	155
linfct_matrix_contrasts	156
list_facades	157
list_to_AnalysisConfiguration	158
list_to_ContrastConfiguration	159
lookup_facade	160
LR_test	161
main_effect_contrasts	162
make_interaction_column	163
make_reduced_hierarchy_config	164
matrix_to_tibble	164
merge_contrasts_results	165
missigness_histogram	166
MissingHelpers	167
missingness_per_condition	169
missingness_per_condition_cumsum	169
Model	170
ModelFirth	172
ModelInterface	175
ModelLimma	176
model_analyse	179
model_summary	180
moderated_p_deqms	181
moderated_p_deqms_long	182
moderated_p_limma	183
multigroup_volcano	184
new_imputed_model	186
nr_B_in_A	187
nr_children_experiment	188
nr_features_experiment	189
nr_obs_sample	190
PACKAGE_DATA	191
pairs_smooth	191
panel_cor	192
pivot_model_contrasts_to_wide	192
plot_estimate	194
plot_heatmap	194
plot_heatmap_cor	196
plot_hierarchies_boxplot	197
plot_hierarchies_boxplot_df	198
plot_hierarchies_line	199

plot_intensity_distribution_violin	200
plot_lrt_diagnostics	201
plot_na_heatmap	201
plot_pca	202
plot_raster	203
plot_sample_correlation	204
plot_stat_density	205
plot_stat_density_median	206
plot_stat_violin	206
plot_stat_violin_median	207
plot_stdv_vs_mean	208
prolfqua_data	209
R6_extract_values	209
rank_peptide_by_intensity	210
register_facade	211
response_matrix_as_tibble	212
robust_scale	213
sample_subset	213
scale_with_subset	214
script_copy_helper_vec	215
separate_hierarchy	215
setup_analysis	216
sigma.rfit_prolfqua	217
sigma.rlm	218
sim_build_models_lm	219
sim_build_models_lmer	220
sim_build_models_logistf	221
sim_lfq_data	222
sim_lfq_data_2factor_config	223
sim_lfq_data_peptide_config	224
sim_lfq_data_protein_config	224
sim_make_model_lm	225
sim_make_model_lmer	226
StrategyBase	227
StrategyBinomial	229
StrategyLimma	230
StrategyLimpa	232
StrategyLM	233
StrategyLmer	235
StrategyLogistf	236
StrategyRfit	238
StrategyRLM	239
strategy_binomial	241
strategy_limma	241
strategy_limpa	242
strategy_logistf	244
summarize_hierarchy	247
summarize_stats	248
summarize_stats_factors	249
summary_ROPECA_median_p_scaled	250
table_facade	252
table_factors	253

table_factors_size	253
tidy_to_wide_config	254
transform_work_intensity	255
upset_interaction_missing_stats	256
upset_missing_stats	257
vcov.rfit_prolfqua	257
volcano_plotly	258
which_missing	259
x5463yzwer453bbb	260

Index	261
--------------	------------

prolfqua-package	<i>prolfqua: Proteomics Label Free Quantification</i>
------------------	---

Description

The prolfqua package integrates essential steps of the mass spectrometry based label free quantification data analysis workflow: quality control, data normalization, protein aggregation, modeling, hypothesis testing and sample size estimation. We use prolfqua to : develop highly customizable, visually appealing, and interactive data analysis reports in pdf or HTML format; to visualize and model simple experimental designs with a single explanatory variable and complex experiments with multiple factors; to benchmark data acquisition, data preprocessing or data modeling methods. We developed and improved the package by applying the "Eating your own dog food" principle, when analysing dozens of proteomics datasets.

Author(s)

Maintainer: Christian Panse <cp@fgcz.ethz.ch> ([ORCID](#))

Authors:

- Witold Wolski <witold.wolski@fgcz.uzh.ch> ([ORCID](#))

Other contributors:

- Functional Genomics Center Zurich [funder]

See Also

Useful links:

- <https://github.com/fgcz/prolfqua>
- <https://fgcz.github.io/prolfqua/>
- Report bugs at <https://github.com/fgcz/prolfqua/issues>

<code>.get_rob scales</code>	<i>compute median and mad on matrix</i>
------------------------------	---

Description

compute median and mad on matrix

Usage

```
.get_rob scales(data, dim = 2)
```

<code>.lfqdata_to_elist</code>	<i>Convert LFQData to limma EList with design matrix and metadata</i>
--------------------------------	---

Description

Shared preamble for all `build_model_limma*` and `build_model_limpa` functions. Pivots LFQ-Data to wide format, builds the design matrix from the formula, resolves the `subject_id` / `isotopeLabel`, and creates a dummy lm for `linfct` extraction.

Usage

```
.lfqdata_to_elist(lfqdata, formula)
```

Arguments

<code>lfqdata</code>	an LFQData object
<code>formula</code>	a formula (with response and RHS)

Value

a list with components:

elist limma EList with `$E` = expression matrix

expr_matrix the expression matrix (same as `elist$E`)

design the design matrix

annotation sample-level annotation data.frame

subject_id character vector of hierarchy keys (possibly including `isotopeLabel`)

rowdata data.frame with one row per feature, columns = `subject_id`

rhs_formula the RHS-only formula

dummy_model a dummy lm fitted on one complete row

<code>.make_progress</code>	<i>Build a progress reporter for the per-protein / per-contrast loops</i>
-----------------------------	---

Description

Internal factory used by `'model_analyse()'` and the contrast loops. The returned object exposes `'$tick(len = 1, tokens = list())'`, the same contract as `'progress::progress_bar'`, so call sites are agnostic to which reporter they received.

Usage

```
.make_progress(
  total,
  label = NULL,
  reporter = getOption("prolfqua.progress", NULL)
)
```

Arguments

<code>total</code>	number of iterations; <code>'<= 0'</code> yields a no-op reporter.
<code>label</code>	short label naming the pass (e.g. "firth multi-peptide").
<code>reporter</code>	reporter selector; see Details. Defaults to <code>'getOption("prolfqua.progress", NULL)'</code> .

Details

The `'reporter'` argument (defaulting to the `'prolfqua.progress'` option) selects the mode:

- `'NULL'` – the legacy `'progress::progress_bar'` (terminal, unchanged).
- a `'function(i, total, label)'` – a throttled user callback, e.g. one that calls `'logger::log_info()'`.
- `'"message"'` – a throttled `'message()'` heartbeat with elapsed/ETA.

Value

an object with a `'$tick()'` method.

<code>.resolve_weights</code>	<i>Resolve strategy weights to a matrix or vector for limma::lmFit</i>
-------------------------------	--

Description

Shared weight resolution logic for `build_model_limma*` functions. Handles character (column name in annotation or LFQData), matrix, or NULL.

Usage

```
.resolve_weights(lfqdata, strategy, annotation)
```

Arguments

lfqdata	an LFQData object
strategy	a strategy object with a weights field
annotation	the sample-level annotation data.frame

Value

a weight matrix, vector, or NULL

adjust_p_values	<i>adjust columns</i>
-----------------	-----------------------

Description

adjust columns

Usage

```
adjust_p_values(
  contrast_df,
  column = "p.value",
  group_by_col = "contrast",
  newname = "FDR"
)
```

Arguments

contrast_df	data.frame with p-values to adjust
column	name of column containing p-values
group_by_col	column(s) to group by before adjusting (e.g. contrast), or NULL for no grouping
newname	name of the new column with adjusted p-values

Value

The computed result.

Examples

```
bb <- c(runif(1000), rexp(1500,rate=5))
length(bb)
bb <- bb[bb < 1]
length(bb)
bb <- bb[1:2000]
hist(bb)
data <- data.frame(contrast = rep(LETTERS[1:5],400), p.value = bb)

dataX <- adjust_p_values(data)
Adata <- dataX |> dplyr::filter(contrast == "A")
stopifnot(all.equal(Adata$FDR, p.adjust(Adata$p.value, method="BH")))
data2 <- adjust_p_values(data, group_by_col = NULL)
stopifnot(all.equal(data2$FDR, p.adjust(data2$p.value, method="BH")))
```

AggregateLimpa	<i>AggregateLimpa</i>
----------------	-----------------------

Description

AggregateLimpa

AggregateLimpa

Details

Aggregates peptide/precursor intensities to protein level using limpa's probabilistic DPC-based quantification (`limpa::dpcQuant`). With `impute_only = TRUE`, performs same-level imputation without aggregation using `limpa::dpcQuantByRow`.

The output `LFQData` contains three value columns:

- intensity ("`limpa`") — protein-level log-expression (complete, no NAs)
- standard error (in `config$opt_se`) — per-protein, per-sample posterior SEs
- observation count (in `config$nr_children`) — number of observed precursors

Value

An R6 class generator.

Super class

`prolfqua::AggregatorBase` -> `AggregateLimpa`

Public fields

`dpc_result` estimated DPC object from `limpa::dpc`

`dpc_slope` DPC slope parameter (default 0.8)

`impute_only` if TRUE use `dpcQuantByRow` (no aggregation)

Methods

Public methods:

- `AggregateLimpa$new()`
- `AggregateLimpa$aggregate()`
- `AggregateLimpa$plot()`
- `AggregateLimpa$write_plots()`
- `AggregateLimpa$clone()`

Method `new()`: initialize

Usage:

```
AggregateLimpa$new(
  lfq,
  prefix = "protein",
  dpc_slope = 0.8,
  impute_only = FALSE
)
```

Arguments:

lfq LFQData with log2-transformed intensities
 prefix default "protein"
 dpc_slope DPC slope parameter passed to limpa (default 0.8)
 impute_only if TRUE, use dpcQuantByRow instead of dpcQuant

Method aggregate(): run limpa DPC-based aggregation (or imputation)

Usage:

```
AggregateLimpa$aggregate()
```

Returns: LFQData

Method plot(): creates aggregation plots (only for aggregation mode, not impute_only)

Usage:

```
AggregateLimpa$plot(subset = NULL, show.legend = FALSE)
```

Arguments:

subset create plots for a subset of the data only
 show.legend default FALSE

Returns: data.frame

Method write_plots(): writes plots to folder

Usage:

```
AggregateLimpa$write_plots(
  qcpath,
  subset = NULL,
  show.legend = FALSE,
  width = 6,
  height = 6
)
```

Arguments:

qcpath qcpath
 subset write plots only for some
 show.legend legend
 width figure width
 height figure height

Returns: file path

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
AggregateLimpa$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other LFQData: [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```

istar <- prolfqua::sim_lfq_data_peptide_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata <- lfqdata$get_Transformer()$log2()$lfq

agg <- AggregateLimpa$new(lfqdata, "protein")
agg$aggregate()
agg$lfq_agg$data_wide()

```

AggregateMedpolish	<i>AggregateMedpolish</i>
--------------------	---------------------------

Description

AggregateMedpolish

AggregateMedpolish

Details

Aggregates peptide intensities to protein level using median polish. Works best with variance-stabilized (log-transformed) intensities.

Value

An R6 class generator.

Super class

`prolfqua::AggregatorBase` -> `AggregateMedpolish`

Methods**Public methods:**

- `AggregateMedpolish$aggregate()`
- `AggregateMedpolish$clone()`

Method `aggregate()`: run median polish aggregation

Usage:

`AggregateMedpolish$aggregate()`

Returns: LFQData

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`AggregateMedpolish$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregatorRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```

istar <- prolfqua::sim_lfq_data_peptide_config()
data <- istar$data |> dplyr::filter(protein_Id %in% sample(protein_Id, 100))
lfqdata <- LFQData$new(data, istar$config)
lfqTrans <- lfqdata$clone()$get_Transformer()$log2()$robscale()$lfq

agg <- AggregateMedpolish$new(lfqTrans, "protein")
agg$aggregate()
p <- agg$plot()
p$plots[[1]]

```

AggregateRlm

*AggregateRlm***Description**

AggregateRlm

AggregateRlm

Details

Aggregates peptide intensities to protein level using robust regression (rlm). Works best with variance-stabilized (log-transformed) intensities.

Value

An R6 class generator.

Super class

`prolfqua::AggregatorBase` -> `AggregateRlm`

Methods**Public methods:**

- `AggregateRlm$aggregate()`
- `AggregateRlm$clone()`

Method `aggregate()`: run robust regression aggregation

Usage:

`AggregateRlm$aggregate()`

Returns: `LFQData`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`AggregateRlm$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()
data <- istar$data |> dplyr::filter(protein_Id %in% sample(protein_Id, 100))
lfqdata <- LFQData$new(data, istar$config)
lfqTrans <- lfqdata$clone()$get_Transformer()$log2()$robscale()$lfq

agg <- AggregateRlm$new(lfqTrans, "protein")
agg$aggregate()
p <- agg$plot()
p$plots[[1]]
```

AggregateTopN	<i>AggregateTopN</i>
---------------	----------------------

Description

AggregateTopN
AggregateTopN

Details

Aggregates peptide intensities to protein level using top N peptides. Works with raw (untransformed) intensities.

Value

An R6 class generator.

Super class

[prolfqua::AggregatorBase](#) -> AggregateTopN

Public fields

N top N peptides by intensity
func aggregation function name: "sum" or "mean"

Methods

Public methods:

- [AggregateTopN\\$new\(\)](#)
- [AggregateTopN\\$aggregate\(\)](#)
- [AggregateTopN\\$clone\(\)](#)

Method new(): initialize

Usage:

```
AggregateTopN$new(lfq, prefix = "protein", N = 3, func = "sum")
```

Arguments:

lfq LFQData

prefix default protein

N top N peptides (default 3)

func "sum" or "mean" (default "sum")

Method aggregate(): run top N aggregation

Usage:

```
AggregateTopN$aggregate()
```

Returns: LFQData

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
AggregateTopN$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()
data <- istar$data |> dplyr::filter(protein_Id %in% sample(protein_Id, 100))
lfqdata <- LFQData$new(data, istar$config)

agg <- AggregateTopN$new(lfqdata, "protein", N = 3, func = "sum")
agg$aggregate()
p <- agg$plot()
p$plots[[1]]

agg_mean <- AggregateTopN$new(lfqdata, "protein", N = 3, func = "mean")
agg_mean$aggregate()
protPlotter <- agg_mean$lfq_agg$get_Plotter()
protPlotter$heatmap()
agg_mean$write_plots(tempdir())
```

aggregate_intensity_top_n

Aggregates top N intensities

Description

run [rank_peptide_by_intensity](#) first

Usage

```
aggregate_intensity_top_n(ranked_data, lfqdata, .func, N = 3)
```

Arguments

ranked_data	data.frame with ranked peptides
lfqdata	LFQData object
.func	function to use for aggregation
N	default 3 top intensities.

Value

list with data and new reduced configuration (config)

See Also

Other aggregation: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [estimate_intensity\(\)](#), [plot_estimate\(\)](#), [plot_hierarchies_line\(\)](#)

Examples

```
dd <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(dd$data, dd$config)
ranked <- rank_peptide_by_intensity(lfq$data_long(), lfq$response(), lfq$hierarchy_keys())

mean_f <- function(x, name = FALSE) {
  if (name) return("mean")
  mean(x, na.rm = TRUE)
}

resTOPN <- aggregate_intensity_top_n(ranked, lfq, .func = mean_f, N = 3)
stopifnot(names(resTOPN) %in% c("data", "config"))
lfq_agg <- LFQData$new(resTOPN$data, resTOPN$config)
tmpRob <- plot_estimate(lfq, lfq_agg, show.legend = TRUE)
stopifnot("ggplot" %in% class(tmpRob$plots[[4]]))
```

AggregatorBase

Base class of the peptide to protein aggregators

Description

Base class of the peptide to protein aggregators

Base class of the peptide to protein aggregators

Details

Holds the fields, construction, and plotting shared by [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#) and [AggregateLimp](#). Subclasses implement `aggregate()`.

Value

An R6 class generator.

Public fields

lfq LFQData
 lfq_agg aggregation result
 prefix to use for aggregation results e.g. protein

Methods**Public methods:**

- [AggregatorBase\\$new\(\)](#)
- [AggregatorBase\\$plot\(\)](#)
- [AggregatorBase\\$write_plots\(\)](#)
- [AggregatorBase\\$clone\(\)](#)

Method new(): initialize

Usage:

```
AggregatorBase$new(lfq, prefix = "protein")
```

Arguments:

lfq LFQData
 prefix default protein

Method plot(): creates aggregation plots

Usage:

```
AggregatorBase$plot(subset = NULL, show.legend = FALSE)
```

Arguments:

subset create plots for a subset of the data only
 show.legend default FALSE

Returns: data.frame

Method write_plots(): writes plots to folder

Usage:

```
AggregatorBase$write_plots(
  qcpath,
  subset = NULL,
  show.legend = FALSE,
  width = 6,
  height = 6
)
```

Arguments:

qcpath qcpath
 subset write plots only for some
 show.legend legend
 width figure width
 height figure height

Returns: file path

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
AggregatorBase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

AnalysisConfiguration *Analysis Configuration*

Description

Analysis Configuration — holds all table annotations, hierarchy definitions, factor definitions, and analysis parameters in a single flat object.

Value

An R6 class generator.

Public fields

`sep` separator to use when uniting columns is necessary

`file_name` column name of column containing raw file names

`sample_name` (will be generated from factors or `file_name`)

`norm_value` optional column with normalization values (e.g., Creatinine)

`isotope_label` which column contains the isotope label (e.g. heavy or light), or light only if LFQ.

`ident_q_value` column name with identification QValues (smaller better)

`ident_score` column with identification score (larger better)

`opt_rt` optional column with rt information

`opt_mz` optional column with mz information

`opt_se` optional column with standard errors (e.g. from limpa aggregation)

`nr_children` optional column containing for instance the number of peptides

`work_intensity` column which contains the intensities

`is_response_transformed` are the intensities transformed for constant variance

`bin_resp` column with encoded missing information

`pattern_decoys` optional regex identifying decoy / reverse-database ids, unioned with the built-in default prefixes (see [is_decoy](#)). NULL (default) leaves decoy handling off; set it (even to "" for defaults only) to opt into flagging and the targets-only model fit.

`pattern_contaminants` optional regex identifying contaminant ids (keratin, trypsin, BSA, ...). NULL (default) means contaminants are not flagged; set it to enable contaminant flagging / removal.

factors Names of columns containing factors (annotations)
factor_depth number of relevant factors (used by plotting functions etc)
hierarchy list with columns describing the measurement hierarchy (i.e. protein peptide precursor fragment)
hierarchy_depth At which depth do you want to model i.e. protein than 1
min_peptides_protein minimum number of peptides per protein

Methods

Public methods:

- `AnalysisConfiguration$set_response()`
- `AnalysisConfiguration$get_response()`
- `AnalysisConfiguration$pop_response()`
- `AnalysisConfiguration$factor_keys()`
- `AnalysisConfiguration$factor_keys_depth()`
- `AnalysisConfiguration$hierarchy_keys()`
- `AnalysisConfiguration$hierarchy_keys_depth()`
- `AnalysisConfiguration$id_vars()`
- `AnalysisConfiguration$value_vars()`
- `AnalysisConfiguration$annotation_vars()`
- `AnalysisConfiguration$clone()`

Method `set_response()`: Add name of intensity column

Usage:

`AnalysisConfiguration$set_response(col_name)`

Arguments:

`col_name` name of intensity column

Method `get_response()`: Get name of working intensity column

Usage:

`AnalysisConfiguration$get_response()`

Method `pop_response()`: Remove last name in array of working intensity column names

Usage:

`AnalysisConfiguration$pop_response()`

Method `factor_keys()`: Get factor keys

Usage:

`AnalysisConfiguration$factor_keys()`

Returns: array with keys

Method `factor_keys_depth()`: Get factor keys till factorDepth

Usage:

`AnalysisConfiguration$factor_keys_depth()`

Method `hierarchy_keys()`: get hierarchy keys

Usage:

AnalysisConfiguration\$hierarchy_keys()

Returns: array of column names

Method hierarchy_keys_depth(): get hierarchy keys up to depth

Usage:

AnalysisConfiguration\$hierarchy_keys_depth(names = TRUE)

Arguments:

names if TRUE names only if FALSE key value pairs

Returns: array of column names

Method id_vars(): get names of columns annotating values (e.g. intensities)

Usage:

AnalysisConfiguration\$id_vars()

Returns: character array

Method value_vars(): get names of columns containing observations e.g. (intensity, qValue, mz or rt)

Usage:

AnalysisConfiguration\$value_vars()

Method annotation_vars(): get names of columns with sample annotations

Usage:

AnalysisConfiguration\$annotation_vars()

Method clone(): The objects of this class are cloneable with this method.

Usage:

AnalysisConfiguration\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

Other configuration: [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
config <- istar$config
stopifnot("AnalysisConfiguration" %in% class(config))
stopifnot(length(config$hierarchy_keys()) > 0)
stopifnot(length(config$factor_keys()) > 0)
stopifnot(length(config$get_response()) == 1)
```

`annotation_add_contrasts`*DRY function: process and export annotated contrasts*

Description

DRY function: process and export annotated contrasts

Usage

```
annotation_add_contrasts(  
  df,  
  primary_col,  
  secondary_col,  
  prefix = "G_",  
  dataset_id = "dataset",  
  decreasing = FALSE,  
  interactions = TRUE  
)
```

Arguments

<code>df</code>	data.frame with annotation data
<code>primary_col</code>	character, name of the primary factor column
<code>secondary_col</code>	character, name of the secondary factor column
<code>prefix</code>	character, prefix for output naming (default "G_")
<code>dataset_id</code>	character, dataset identifier for output naming
<code>decreasing</code>	logical, if TRUE sort factor levels in decreasing order
<code>interactions</code>	logical, if TRUE include interaction contrasts

Value

Contrast definitions or contrast results.

See Also

Other contrasts: [generate_contrasts\(\)](#), [generate_contrasts_for_factor\(\)](#), [interaction_contrasts\(\)](#), [level_specific_contrasts\(\)](#), [main_effect_contrasts\(\)](#)

Examples

```
data("x5463yzwer453bbb", package = "prolfqua")  
annotation_add_contrasts(x5463yzwer453bbb, "factor_A", "factor_B", "primary")  
annotation_add_contrasts(x5463yzwer453bbb, "factor_B", "factor_A", "secondary")
```

AnovaExtractor

R6 class for extracting ANOVA results as a data frame

Description

R6 class for extracting ANOVA results as a data frame

R6 class for extracting ANOVA results as a data frame

Details

Wraps the ANOVA extraction logic and associated column names.

Value

An R6 class generator.

Public fields

test statistical test type ("F", "Chisq", etc.)

col_pval p-value column name in ANOVA output

Methods

Public methods:

- [AnovaExtractor\\$new\(\)](#)
- [AnovaExtractor\\$fun\(\)](#)
- [AnovaExtractor\\$clone\(\)](#)

Method new(): Create a new AnovaExtractor

Usage:

```
AnovaExtractor$new(test = "F")
```

Arguments:

test statistical test type

Method fun(): Extract ANOVA table from a fitted model

Usage:

```
AnovaExtractor$fun(x)
```

Arguments:

x a fitted model

Returns: data.frame with factor column and ANOVA statistics

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
AnovaExtractor$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_2\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
extractor <- AnovaExtractor$new(test = "F")
fit <- stats::lm(Sepal.Length ~ Species, data = iris)
res <- extractor$fun(fit)
stopifnot("factor" %in% colnames(res))
```

`apply_to_response_matrix`

Apply function requiring a matrix to tidy table

Description

Apply function requiring a matrix to tidy table

Usage

```
apply_to_response_matrix(lfqdata, .func, .funcname = NULL)
```

Arguments

<code>lfqdata</code>	LFQData object
<code>.func</code>	function taking and returning a matrix
<code>.funcname</code>	name of function (used for creating new column)

Value

list with 'data' (data.frame) and 'colname' (new column name)

See Also

Other preprocessing: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [filter_proteins_by_peptide_count\(\)](#), [get_rob scales\(\)](#), [nr_B_in_A\(\)](#), [robust_scale\(\)](#), [scale_with_subset\(\)](#)

Examples

```
bb <- sim_lfq_data_peptide_config(Nprot = 100)
lfqdata <- LFQData$new(bb$data, bb$config)
res <- apply_to_response_matrix(lfqdata, .func = base::scale)
stopifnot("abundance_base..scale" %in% colnames(res$data))

res <- apply_to_response_matrix(lfqdata, .func = robust_scale)

if (require("vsn")) {
  res <- apply_to_response_matrix(lfqdata, .func = vsn::justvsn)
}
```

build_contrast_analysis

Build a contrast analysis using one of several statistical methods

Description

A builder function that dispatches to the appropriate facade class based on the chosen method. Each facade encapsulates the full pipeline from strategy construction through modelling to contrast computation.

Usage

```
build_contrast_analysis(lfqdata, modelstr, contrasts, method = "lm", ...)
```

Arguments

lfqdata	an LFQData object
modelstr	model formula string without the response variable (e.g. " <code>~ group_</code> "). The response is taken automatically from <code>lfqdata\$get_config()</code> <code>\$get_response()</code> .
contrasts	named character vector of contrasts (e.g. <code>c("A_vs_B" = "group_A - group_B")</code>)
method	a registered facade key. The built-in keys are " <code>lm</code> ", " <code>lm_impute</code> ", " <code>limma</code> ", " <code>limma_impute</code> ", " <code>limma_voom</code> ", " <code>limma_voom_impute</code> ", " <code>limpa</code> ", " <code>limpa_nested</code> ", " <code>rlm</code> ", " <code>rfit</code> ", " <code>rfit_impute</code> ", " <code>deqms</code> ", " <code>deqms_voom</code> ", " <code>firth</code> ", " <code>firth_nested</code> ", " <code>binomial_nested</code> ", " <code>lmer_nested</code> ", " <code>ropeca_nested</code> "; downstream packages may add more via register_facade . The authoritative list is <code>names(list_facades())</code> . Defaults to " <code>lm</code> ".
...	additional arguments forwarded to the underlying strategy function (e.g. <code>trend</code> , <code>robust</code> for <code>strategy_limma</code>)

Value

one of [ContrastsLimmaFacade](#), [ContrastsLMFacade](#), [ContrastsRLMFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsLMImputeFacade](#), [ContrastsDEqMSFacade](#), [ContrastsROPECANestedFacade](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLimpaFacade](#), [ContrastsBinomialNestedFacade](#), or [ContrastsLimpaNestedFacade](#)

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 20)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")

fa_lm <- build_contrast_analysis(lfqdata, "~ group_", contrasts, method = "lm")
head(fa_lm$get_contrasts())

fa_limma <- build_contrast_analysis(lfqdata, "~ group_", contrasts, method = "limma")
head(fa_limma$get_contrasts())

fa_deqms <- build_contrast_analysis(lfqdata, "~ group_", contrasts, method = "deqms")
head(fa_deqms$get_contrasts())

istar_pep <- sim_lfq_data_peptide_config()
lfqdata_pep <- LFQData$new(istar_pep$data, istar_pep$config)
lfqdata_pep <- lfqdata_pep$get_Transformer()$log2()$lfq

fa_lmer <- build_contrast_analysis(
  lfqdata_pep,
  "~ group_ + (1 | peptide_Id) + (1 | sampleName)",
  contrasts,
  method = "lmer_nested"
```

```

)
head(fa_lmer$get_contrasts())

fa_ropeca <- build_contrast_analysis(lfqdata_pep, "~ group_", contrasts, method = "ropeca_nested")
head(fa_ropeca$get_contrasts())

fa_firth <- build_contrast_analysis(lfqdata, "~ group_", contrasts, method = "firth")
head(fa_firth$get_contrasts())

```

build_model

Build protein models from data

Description

Build protein models from data

Usage

```

build_model(
  data,
  model_strategy,
  subject_id = if ("LFQData" %in% class(data)) {
    data$subject_id()
  } else {
    "protein_Id"
  },
  model_name = model_strategy$model_name
)

```

Arguments

data	data - a data frame or LFQData object
model_strategy	model strategy object (e.g. from strategy_lmer or strategy_lm)
subject_id	grouping variable
model_name	model name

Value

a object of class `Model`

See Also

[model_analyse](#), [strategy_lmer](#) [strategy_lm](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#),

```

Model, ModelFirth, ModelLimma, StrategyLM, StrategyLimma, StrategyLimpa, StrategyLmer,
StrategyLogistf, StrategyRLM, StrategyRfit, build_contrast_analysis(), build_model_glm_peptide(),
build_model_glm_protein(), build_model_impute(), build_model_limma(), build_model_limma_impute(),
build_model_limma_voom(), build_model_limma_voom_impute(), build_model_limpa(), build_model_logistf(),
compute_borrowed_variance(), compute_borrowed_variance_limma(), compute_contrast(),
compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(),
get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(),
linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_res,
model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(),
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()

```

Examples

```

D <- prolfqua::sim_1fq_data_peptide_config(Nprot = 20, weight_missing = 0.1)
D$data$abundance |> is.na() |> sum()
D <- prolfqua::sim_1fq_data_peptide_config(Nprot = 20, weight_missing = 0.1, seed = 3)
D$data$abundance |> is.na() |> sum()
model_name <- "f_condition_r_peptide"
formula_randomPeptide <-
  strategy_lmer("abundance ~ group_ + (1 | peptide_Id) + (1 | sampleName)",
    model_name = model_name)

mod <- prolfqua::build_model(
  D$data,
  formula_randomPeptide,
  model_name = model_name,
  subject_id = D$config$hierarchy_keys_depth())
aovtable <- mod$get_anova()

mod <- prolfqua::build_model(
  LFQData$new(D$data, D$config),
  formula_randomPeptide,
  model_name = model_name)
model_summary(mod)

```

build_model_glm_peptide

Build Firth logistic model for nested LFQData

Description

Encodes missingness as a binary response and fits the peptide-aware Firth logistic backend. Proteins with multiple child features are fitted with the lowest hierarchy key appended to the formula.

Usage

```
build_model_glm_peptide(lfqdata, modelstr)
```

Arguments

lfqdata nested [LFQData](#) object
 modelstr model formula string without the response variable (e.g. "~ group_")

Value

a [ModelFirth](#) object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config(
  Nprot = 10, with_missing = TRUE, weight_missing = 0.5, seed = 3)
lfqdata <- LFQData$new(istar$data, istar$config)
mod <- build_model_glm_peptide(lfqdata, "~ group_")
head(mod$get_coefficients())
```

build_model_glm_protein

Build Firth logistic model for aggregated LFQData

Description

Encodes missingness as a binary response and fits the Firth logistic backend used by the missingness model path in prolfquapp.

Usage

```
build_model_glm_protein(lfqdata, modelstr)
```

Arguments

lfqdata	aggregated LFQData object
modelstr	model formula string without the response variable (e.g. "~ group_")

Value

a [ModelFirth](#) object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_residues\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(
  Nprot = 10, with_missing = TRUE, weight_missing = 0.5, seed = 3)
lfqdata <- LFQData$new(istar$data, istar$config)
mod <- build_model_glm_protein(lfqdata, "~ group_")
head(mod$get_coefficients())
```

build_model_impute	<i>Build protein models with LOD imputation for failed fits</i>
--------------------	---

Description

Fits per-protein models, then re-fits failed/singular proteins after imputing missing values with the limit of detection (LOD) and clamping. Covariance is borrowed from successful fits so that variance is not underestimated by the constant imputation.

Usage

```
build_model_impute(
  lfqdata,
  model_strategy,
  model_name = paste0(model_strategy$model_name, "Imputed"),
  lod = NULL,
  borrow_method = c("sigma", "vcov"),
  df_method = c("observed", "borrowed"),
  on_misalign = c("fallback", "fail")
)
```

Arguments

lfqdata	LFQData object (aggregated to protein level)
model_strategy	model strategy object (e.g. from strategy_lm)
model_name	model name (default appends "Imputed")
lod	numeric limit of detection; if NULL, auto-computed from data
borrow_method	"sigma" borrows scalar sigma and uses per-protein $(X'X)^{-1}$; "vcov" borrows element-wise median of full vcov matrices
df_method	"observed" uses $\max(n_{\text{observed}} - p, 1)$; "borrowed" uses median df from successful fits
on_misalign	passed to compute_borrowed_variance via impute_refit_singular . "fallback" (default) preserves the lm behaviour; "fail" skips the rescue when a full covariance matrix cannot be borrowed (used by the rfit backend, which has no sigma-method fallback).

Value

a object of class [Model](#)

See Also

[build_model](#), [impute_refit_singular](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#),

```
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()
```

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 30, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
strat <- strategy_lm(paste(lfqdata$response(), "~ group"))
mod <- build_model_impute(lfqdata, strat)
```

build_model_limma	<i>Build limma model from LFQData</i>
-------------------	---------------------------------------

Description

Analogous to [build_model](#) but uses limma's matrix-based pipeline. Takes an [LFQData](#) object and a strategy from [strategy_limma](#), pivots data to wide format, fits with [lmFit](#), and returns a [Modellimma](#) object.

Usage

```
build_model_limma(lfqdata, strategy, model_name = strategy$model_name)
```

Arguments

lfqdata	an LFQData object
strategy	output of strategy_limma
model_name	name of model (default from strategy)

Value

a [Modellimma](#) object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [Modellimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#),

```

linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_res
model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(),
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()

```

Examples

```

istar <- sim_lfq_data_protein_config(Nprot = 50)
lProt <- LFQData$new(istar$data, istar$config)
lProt$rename_response("transformedIntensity")

strat <- strategy_limma("transformedIntensity ~ group_")
mod_limma <- build_model_limma(lProt, strat)
mod_limma$get_coefficients()
mod_limma$get_anova()

```

build_model_limma_impute

Build limma model with LOD imputation for failed proteins

Description

Analogous to [build_model_impute](#) but for limma's matrix-based pipeline. Fits all proteins with [lmFit](#), identifies proteins with NA coefficients (typically from entire missing groups), imputes their missing values with the limit of detection (LOD), refits, and replaces the variance with a borrowed estimate from successful proteins.

Usage

```

build_model_limma_impute(
  lfqdata,
  strategy,
  model_name = paste0(strategy$model_name, "Imputed"),
  lod = NULL,
  df_method = c("observed", "borrowed")
)

```

Arguments

lfqdata	an LFQData object (aggregated to protein level)
strategy	output of strategy_limma
model_name	name of model (default: strategy name + "Imputed")
lod	numeric limit of detection; if NULL, auto-computed from data via MissingHelpers
df_method	how to set degrees of freedom for imputed proteins: "observed" (default) uses $\max(n_{\text{observed}} - p, 1)$ where n_{observed} counts only non-missing values; "borrowed" uses the median df from successful fits

Details

The LOD imputation gives plausible coefficients (fold change direction), while the borrowed sigma and corrected degrees of freedom ensure that inference is not artificially precise from the constant imputation.

Value

a [ModelLimma](#) object with a hybrid fit

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")

strat <- strategy_limma("transformedIntensity ~ group_")
mod <- build_model_limma_impute(lfqdata, strat)
mod$get_coefficients()
```

```
build_model_limma_voom
```

Build limma model with vooma precision weights (proteomics)

Description

Estimates observation-level precision weights from a mean-variance trend (vooma) and fits a weighted least squares model via [lmFit](#). For proteomics data that is already log2-transformed.

Usage

```
build_model_limma_voom(
  lfqdata,
  strategy,
  model_name = strategy$model_name,
  span = 0.5,
  plot = FALSE
)
```

Arguments

lfqdata	an LFQData object (aggregated to protein level)
strategy	output of strategy_limma
model_name	name of model (default from strategy)
span	lowess smoother span for mean-variance trend (default 0.5)
plot	logical; if TRUE, plot the mean-variance trend

Details

When `strategy$weights` is set (e.g. to `nr_children`), these external weights enter the preliminary fit (so the trend estimation accounts for measurement precision) and are multiplied element-wise with the vooma weights for the final fit. See the voom integration notes in `../TODO/prolfqua/Archive/TTODO_limma_voom` for the mathematical basis.

Value

a [ModelLimma](#) object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50)
lProt <- LFQData$new(istar$data, istar$config)
lProt$rename_response("transformedIntensity")

strat <- strategy_limma("transformedIntensity ~ group_")
mod <- build_model_limma_voom(lProt, strat)
mod$get_coefficients()
```

```
build_model_limma_voom_impute
```

Build limma-voom model with LOD imputation for failed proteins

Description

Combines vooma precision weights with LOD imputation for proteins that have entire missing groups (NA coefficients). Mirrors [build_model_limma_impute](#) but uses vooma weights.

Usage

```
build_model_limma_voom_impute(
  lfqdata,
  strategy,
  model_name = paste0(strategy$model_name, "Imputed"),
  lod = NULL,
  df_method = c("observed", "borrowed"),
  span = 0.5,
  plot = FALSE
)
```

Arguments

lfqdata	an LFQData object (aggregated to protein level)
strategy	output of strategy_limma
model_name	name of model (default: strategy name + "Imputed")
lod	numeric limit of detection; if NULL, auto-computed from data
df_method	how to set degrees of freedom for imputed proteins: "observed" (default) uses $\max(n_{\text{observed}} - p, 1)$; "borrowed" uses the median df from successful fits
span	lowess smoother span for vooma trend (default 0.5)
plot	logical; if TRUE, plot the mean-variance trend

Value

a [ModelLimma](#) object with a hybrid fit

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")

strat <- strategy_limma("transformedIntensity ~ group_")
mod <- build_model_limma_voom_impute(lfqdata, strat)
mod$get_coefficients()
```

build_model_limpa	<i>Build limpa vooma model from aggregated LFQData</i>
-------------------	--

Description

Takes protein-level LFQData produced by [AggregateLimpa](#) (which contains standard error and observation count columns) and fits a limpa vooma model via `limpa::voomaLmFitWithImputation`. The standard errors from DPC quantification are used as a bivariate predictor for the vooma precision weights, and the observation counts identify imputed entries for degree-of-freedom correction.

Usage

```
build_model_limpa(lfqdata, strategy, model_name = strategy$model_name)
```

Arguments

lfqdata	LFQData object. Must have <code>config\$nr_children</code> set (used to flag imputed observations via <code>nr_children == 0</code>). <code>config\$opt_se</code> is optional: when set (e.g. from AggregateLimpa), the SE column is used as a vooma precision-weight predictor; otherwise vooma is fit without an external predictor.
---------	--

strategy	StrategyLimpa object
model_name	model name (default from strategy)

Details

Returns a [ModelLimma](#) object (since the output is a standard MArrayLM), which can be passed directly to [ContrastsLimma](#).

Value

[ModelLimma](#) object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
if (requireNamespace("limpa", quietly = TRUE)) {
  istar <- sim_lfq_data_peptide_config(Nprot = 10)
  lfq <- LFQData$new(istar$data, istar$config)
  lfq <- lfq$get_Transformer()$log2()$lfq
  lfq <- AggregateLimpa$new(lfq, "protein")$aggregate()
  strat <- strategy_limpa("limpa ~ group_")
  mod <- build_model_limpa(lfq, strat)
  mod$model_name
}
```

build_model_logistf	<i>build_model_logistf</i>
---------------------	----------------------------

Description

build_model_logistf

Usage

```
build_model_logistf(data, formula)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config(Nprot = 10, with_missing = TRUE,
  weight_missing = 0.5, seed = 3)
istar$data <- prolfqua::encode_bin_resp(LFQData$new(istar$data, istar$config))
istar$config$bin_resp <- "bin_resp"
tmp <- LFQData$new(istar$data, istar$config)
formula <- paste0(tmp$get_config()$bin_resp , "~ group_")
xx2 <- build_model_logistf(tmp, formula)

istar <- prolfqua::sim_lfq_data_protein_config(Nprot = 10, with_missing = TRUE,
  weight_missing = 0.5, seed = 3)
istar$data <- prolfqua::encode_bin_resp(LFQData$new(istar$data, istar$config))
istar$config$bin_resp <- "bin_resp"
tmp <- LFQData$new(istar$data, istar$config)
formula <- paste0(tmp$get_config()$bin_resp , "~ group_")
xx <- build_model_logistf(tmp, formula)
```

```

contrasts <- c(AvsB = "group_A - group_B")
m <- xx$models$models1$model_df$linear_model[[1]]
linfct_matrix_contrasts(linfct_from_model(m)$linfct_factors, contrasts)
m <- xx2$models$models2$model_df$linear_model[[1]]
linfct_matrix_contrasts(linfct_from_model(m)$linfct_factors, contrasts)

```

center_to_reference_cfg

center to reference

Description

takes the median of the lfqdata reference per sample and subtracts it from a copy of lfqdata

Usage

```
center_to_reference_cfg(lfqdata, lfqdatareference)
```

Arguments

lfqdata LFQData object containing the data to center
lfqdatareference LFQData object containing the reference subset

Value

The computed result.

Examples

```

# example code

bb <- sim_lfq_data_peptide_config(Nprot = 100)
x <- LFQData$new(bb$data, bb$config)
xc <- x$get_copy()
xc$set_data(xc$data_long() |> dplyr::filter(protein_Id == "0EfVhX~3967"))
xxd <- center_to_reference_cfg(x, xc)
xxd$response()
x$response()

```

complete_cases

Complete cases

Description

The tidy table does not need to contain missing data. This function re-establishes the missing observations in a sample.

Usage

```
complete_cases(lfqdata)
```

Arguments

lfqdata LFQData object

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hiera](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
bb <- sim_lfq_data_protein_config()
lfq <- LFQData$new(bb$data, bb$config)
xx <- complete_cases(lfq)
stopifnot(nrow(bb$data) <= nrow(xx))
```

compute_borrowed_variance

Compute borrowed variance from successful model fits

Description

Compute borrowed variance from successful model fits

Usage

```
compute_borrowed_variance(
  model_df,
  method = c("sigma", "vcov"),
  on_misalign = c("fallback", "fail")
)
```

Arguments

model_df	tibble from model_analyse
method	"sigma" borrows scalar sigma and uses per-protein $(X'X)^{-1}$, "vcov" borrows element-wise median of full vcov matrices
on_misalign	behaviour when donor vcov matrices cannot be aligned by coefficient name (method = "vcov" only). "fallback" (default) warns and falls back to the sigma method, preserving the lm behaviour. "fail" signals a <code>prolfqua_borrow_failed</code> condition so the caller can skip the rescue; backends without an <code>lm</code> -style <code>summary(fit)\$cov.unscaled</code> (e.g. <code>rfit</code>) must use this because the sigma fallback is not applicable to them.

Value

list with sigma, df, method, and optionally vcov

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
mod <- sim_build_models_lm(model = "parallel3", weight_missing = 1)

# Sigma method (default): returns median sigma and df from donors
borrowed_s <- prolfqua::compute_borrowed_variance(
  mod$model_df, method = "sigma")
stopifnot(borrowed_s$method == "sigma")
stopifnot(is.numeric(borrowed_s$sigma) && borrowed_s$sigma > 0)
stopifnot(is.numeric(borrowed_s$df) && borrowed_s$df > 0)

# Vcov method: element-wise median vcov from donors.
# Falls back to sigma if donor models have misaligned coefficient names.
mod_no_missing <- sim_build_models_lm(model = "parallel3",
  Nprot = 10, with_missing = FALSE)
borrowed_v <- prolfqua::compute_borrowed_variance(
  mod_no_missing$model_df, method = "vcov")
stopifnot(borrowed_v$method == "vcov")
stopifnot(is.matrix(borrowed_v$vcov))
```

compute_borrowed_variance_limma

Compute borrowed variance from successful limma fits

Description

Extracts median sigma and df from proteins that fitted successfully (no NA coefficients). Used by [build_model_limma_impute](#) to replace the artificially low variance of LOD-imputed proteins.

Usage

```
compute_borrowed_variance_limma(fit)
```

Arguments

fit MArrayLM object from `lmFit`

Value

list with sigma (median residual SD) and df (median residual df) from successful proteins

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_limma\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

compute_contrast	<i>handles incomplete models</i>
------------------	----------------------------------

Description

only keeps non NA coefficients.

Usage

```
compute_contrast(m, linfct, confint = 0.95)
```

Arguments

m linear model generated using `lm`
linfct linear function
confint confidence interval default 0.95

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_robust\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_residues\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
m <- sim_make_model_lm( "factors")
linfct <- linfct_from_model(m)$linfct_factors
compute_contrast(m, linfct, confint = 0.95)
compute_contrast(m, linfct, confint = 0.99)
```

`compute_lmer_contrast` *applies contrast computation using lmerTest::contest function*

Description

applies contrast computation using lmerTest::contest function

Usage

```
compute_lmer_contrast(model, linfct, ddf = c("Satterthwaite", "Kenward-Roger"))
```

Arguments

<code>model</code>	mixed effects model
<code>linfct</code>	linear function
<code>ddf</code>	method to determine denominator degrees of freedom

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_2\(\)](#), [compute_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_residues\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
mb <- sim_make_model_lmer("interaction")
summary(mb)

linfct <- linfct_from_model(mb)
names(linfct)
compute_lmer_contrast(mb, linfct$linfct_factors)
compute_lmer_contrast(mb, linfct$linfct_interactions)
length(lme4::fixef(mb))
lmerTest::contest(mb, c( 0 , 1 , 0 , 0 ), joint = FALSE)
summary(mb)
```

compute_pooled	<i>compute pooled variance</i>
----------------	--------------------------------

Description

following the documentation here: <https://online.stat.psu.edu/stat500/lesson/7/7.3/7.3.1/7.3.1.1>

Usage

```
compute_pooled(x)

poolvar(res1, config)
```

Arguments

x data.frame

Value

data.frame

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
x <- data.frame(nrMeasured = c(1,2,2), var = c(3,4,4), meanAbundance = c(3,3,3))
x <- data.frame(nrMeasured = c(1,2,1,1), var = c(NA, 0.0370, NA, NA),
  meanAbundance = c(-1.94,-1.46,-1.87,-1.45))
compute_pooled(x)
y <- data.frame(dilution=c("a","b","c"),
  nrReplicates = c(4,4,4), nrMeasured = c(0,0,1), sd = c(NA,NA,NA),
  var = c(NA,NA,NA), meanAbundance = c(NaN,NaN,NaN))
compute_pooled(y)
yb <- y |> dplyr::filter(nrMeasured > 1)

bb <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb$data, bb$config)
res1 <- summarize_stats(lfq)
pv <- poolvar(res1, bb$config)
stopifnot(nrow(pv) == nrow(res1) / 3)
```

ContrastConfiguration *Contrast Configuration*

Description

Holds the column-role mapping and behaviour flags that describe how a specific contrast backend (lm, limma, SAINT, ...) lays out its result table. Mirrors [AnalysisConfiguration](#) for the modelling side: it carries names, not data.

Consumers that need to operate on a contrast table generically (filter, build ORA inputs, build rank lists, summarise per protein) should read column names through the accessors on this object rather than hard-coding them. This lets backends with different column conventions (e.g. SAINT's BFDR/log2_EFCs) plug into the same downstream code path as LM-style backends without renaming columns.

Defaults match the standard prolfqua contrast schema produced by [Contrasts](#), [ContrastsLimma](#), [ContrastsModerated](#) etc.

Value

An R6 class generator.

Public fields

`subject_id` columns identifying the contrast subject (e.g. protein id)
`model_name_col` column carrying the model name label
`contrast_col` column carrying the contrast label
`effect_col` column with the (signed) effect size; for LM this is `diff`, for SAINT `log2_EFCs`
`score_col` column with the per-contrast test statistic
`pvalue_col` column with the raw p-value, or "" for backends that do not produce one (e.g. SAINTexpress). `NA_character_` is accepted by the constructor and stored as "".
`fdr_col` column with the FDR / BFDR / adjusted p-value
`avg_abundance_col` column with the per-feature mean abundance
`supports_dea_qc` whether the backend supports the legacy differential-expression QC HTML report
`needs_saint_annotation` whether the backend needs annotation reading in SAINT mode (bait / control columns)
`significance_directional` if TRUE, the backend only considers positive-effect features biologically meaningful (e.g. SAINTexpress, where negative `log2_EFCs` are not interpreted as "down" hits). Affects `filter_significant()`: when TRUE the filter is one-sided on the effect column; when FALSE it is symmetric (`|effect| > threshold`).

Methods**Public methods:**

- [ContrastConfiguration\\$new\(\)](#)
- [ContrastConfiguration\\$has_pvalue\(\)](#)
- [ContrastConfiguration\\$clone\(\)](#)

Method `new()`: Construct a `ContrastConfiguration`. All arguments map directly to fields of the same name.

Usage:

```

ContrastConfiguration$new(
  subject_id = character(),
  model_name_col = "modelName",
  contrast_col = "contrast",
  effect_col = "diff",
  score_col = "statistic",
  pvalue_col = "p.value",
  fdr_col = "FDR",
  avg_abundance_col = "avgAbd",
  supports_dea_qc = TRUE,
  needs_saint_annotation = FALSE,
  significance_directional = FALSE
)

```

Arguments:

`subject_id` columns identifying the contrast subject
`model_name_col` model-name column
`contrast_col` contrast-label column
`effect_col` signed effect column

score_col test-statistic column
 pvalue_col raw p-value column, or "" / NA_character_ when the backend produces none
 fdr_col FDR / adjusted-p column
 avg_abundance_col mean-abundance column
 supports_dea_qc supports DEA QC HTML
 needs_saint_annotation needs SAINT-mode annotation
 significance_directional one-sided filter on effect column

Method has_pvalue(): Does the backend produce a usable p-value column?

Usage:

```
ContrastConfiguration$has_pvalue()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ContrastConfiguration$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other configuration: [AnalysisConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```

cfg <- ContrastConfiguration$new(subject_id = "protein_Id")
cfg$effect_col
cfg$fdr_col
cfg$has_pvalue()
cfg$supports_dea_qc

# SAINT-flavoured config
saint <- ContrastConfiguration$new(
  subject_id = "protein_Id",
  contrast_col = "Bait",
  effect_col = "log2_EFCs",
  score_col = "SaintScore",
  pvalue_col = NA_character_,
  fdr_col = "BFDR",
  supports_dea_qc = FALSE,
  needs_saint_annotation = TRUE
)
saint$has_pvalue()

```

Contrasts

*Estimate contrasts using Wald Test***Description**

Estimate contrasts using Wald Test

Estimate contrasts using Wald Test

Details

The per-protein contrast computation uses `compute_contrast` and `linfct_matrix_contrasts` internally.

Value

An R6 class generator.

Super class

`prolfqua::ContrastsInterface` -> Contrast

Public fields

`models` Model

`contrasts` character with contrasts

`contrastfun` function to compute contrasts

`model_name` model name

`subject_id` name of column containing e.g., protein Id's

`p.adjust` function to adjust p-values (default `prolfqua::adjust_p_values`)

`contrast_result` data frame containing results of contrast computation

`global` use a global linear function (determined by `get_linfct`)

Methods**Public methods:**

- `Contrasts$new()`
- `Contrasts$get_contrast_sides()`
- `Contrasts$get_linfct()`
- `Contrasts$get_contrasts()`
- `Contrasts$clone()`

Method `new()`: initialize create Contrast

Usage:

```
Contrasts$new(
  model,
  contrasts,
  p.adjust = prolfqua::adjust_p_values,
  global = FALSE,
  model_name = "WaldTest"
)
```

Arguments:

`model` a dataframe with a structure similar to that generated by [build_model](#)
`contrasts` a character vector with contrast specification
`p.adjust` function to adjust the p-values
`global` development/internal argument (if FALSE determine linfct for each model.)
`model_name` name of contrast method, default WaldTest

Method `get_contrast_sides()`: get both sides of contrasts

Usage:

```
Contrasts$get_contrast_sides()
```

Method `get_linfct()`: get linear functions from contrasts

Usage:

```
Contrasts$get_linfct(global = TRUE, avg = TRUE)
```

Arguments:

`global` logical TRUE - get the a linear functions for all models, FALSE - linear function for each model
`avg` logical TRUE - get also linfct for averages

Method `get_contrasts()`: get table with contrast estimates

Usage:

```
Contrasts$get_contrasts(all = FALSE)
```

Arguments:

`all` should all columns be returned (default FALSE)

Returns: data.frame with contrasts

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Contrasts$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNested](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#),

```

impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(),
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(),
register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(),
sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(),
strategy_limpa(), strategy_logistf(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

```

Examples

```

# Fitting mixed effects model to peptide data
istar <- prolfqua::sim_lfq_data_peptide_config()

modelFunction <-
  strategy_lmer("abundance ~ group_ + (1 | peptide_Id) + (1 | sample)")

config <- istar$config
config$hierarchy_keys_depth()

mod <- build_model(
  istar$data,
  modelFunction,
  subject_id = config$hierarchy_keys_depth()
)

ref_lfc <- data.frame(
  `(Intercept)` = c(0, 0, 0),
  group_B = c(0, 1, 0),
  group_Ctrl = c(-1, -1, 1),
  row.names = c("groupA_vs_Ctrl", "dil.e_vs_b", "dil.ctrl_vs_b")
)
prolfqua::model_summary(mod)
Contr <- c(
  "groupA_vs_Ctrl" = "group_A - group_Ctrl",
  "dil.e_vs_b" = "group_B - group_Ctrl",
  "dil.ctrl_vs_b" = "group_Ctrl - group_A"
)
contrastX <- prolfqua::Contrasts$new(mod, Contr)
y <- contrastX$get_linfct(avg = FALSE)
stopifnot(all(ref_lfc == y))
t <- contrastX$get_linfct(global = FALSE)

x <- contrastX$get_contrasts()
stopifnot(all(x$p.value < 1 & x$p.value > 0))
stopifnot(all(x$avgAbd > 0))
stopifnot(all(x$FDR > 0 & x$FDR < 1))

x <- contrastX$get_contrast_sides()
xd <- contrastX$column_description()
modelFunction <-
  strategy_lm("abundance ~ group_")
mod <- build_model(
  istar$data,
  modelFunction,
  subject_id = config$hierarchy_keys_depth()
)
contrastX <- prolfqua::Contrasts$new(mod, Contr)
y <- contrastX$get_linfct(avg = FALSE)

```

```
stopifnot(all(ref_lfc == y))
```

ContrastsBinomialNestedFacade

Quasibinomial detection-count facade for nested input

Description

Quasibinomial detection-count facade for nested input

Quasibinomial detection-count facade for nested input

Details

Completes and encodes child-feature detection using the same preparation as [ContrastsFirthNestedFacade](#), then collapses the binary rows into detected and undetected counts per parent and sample. The resulting quasibinomial model reports parent-level log odds ratios in diff; avgAbd is the average linear predictor on the log-odds scale.

Child features are treated as exchangeable binomial trials. The symmetric pseudo-count stabilizes complete separation but is not equivalent to Firth's bias-reducing penalty. Empirical-Bayes dispersion moderation uses [ContrastsModerated](#); by default posterior dispersion is bounded below by one.

Value

An R6 class generator.

Super classes

```
prolfqua::ContrastsInterface -> prolfqua::ContrastsFacadeBase -> ContrastsBinomialNestedFacade
```

Methods

Public methods:

- [ContrastsBinomialNestedFacade\\$new\(\)](#)
- [ContrastsBinomialNestedFacade\\$clone\(\)](#)

Method `new()`: Fit the nested detection-count model and its contrasts.

Usage:

```
ContrastsBinomialNestedFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  prior_count = 0.1,
  binomial_bound = TRUE,
  ...
)
```

Arguments:

lfqdata nested [LFQData](#)

`modelstr` right-hand-side model formula
`contrasts` named contrast expressions
`prior_count` non-negative symmetric pseudo-count
`binomial_bound` bound posterior dispersion below by one
`...` passed to [strategy_binomial](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsBinomialNestedFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```

istar <- sim_lfq_data_peptide_config(Nprot = 20, weight_missing = 0.5, seed = 3)
lfqdata <- LFQData$new(istar$data, istar$config)
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
facade <- ContrastsBinomialNestedFacade$new(lfqdata, "~ group_", contrasts)
head(facade$get_contrasts())

```

ContrastsDEqMSFacade *DEqMS contrast analysis facade*

Description

DEqMS contrast analysis facade

DEqMS contrast analysis facade

Details

Encapsulates the pipeline: [strategy_lm](#) -> [build_model](#) -> [Contrasts](#) -> [ContrastsModeratedDEqMS](#).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsDEqMSFacade](#)

Methods

Public methods:

- [ContrastsDEqMSFacade\\$new\(\)](#)
- [ContrastsDEqMSFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsDEqMSFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  weights = lfqdata$nr_children_col(),
  ...
)
```

Arguments:

`lfqdata` LFQData object
`modelstr` model formula string (e.g. "~ group_")
`contrasts` named character vector of contrasts
`weights` column name for per-observation weights (default: `lfqdata$nr_children_col()`).
 Pass NULL for unweighted.
 ... passed to `strategy_lm`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsDEqMSFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsDEqMSFacade$new(lfqdata, "~ group_", contrasts)
```

```
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsDEqMSVoomFacade

DEqMS contrast analysis with vooma weights facade

Description

DEqMS contrast analysis with vooma weights facade

DEqMS contrast analysis with vooma weights facade

Details

Combines vooma precision weights (mean-variance trend) with DEqMS count-dependent variance moderation. Vooma runs **without** external weights so it captures only the mean-variance relationship; the peptide count enters solely through DEqMS moderation, avoiding double-counting.

Pipeline: [strategy_limma](#) (weights = NULL) -> [build_model_limma_voom](#) -> [ContrastsLimma](#) (eBayes = FALSE) -> [ContrastsModeratedDEqMS](#).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsDEqMSVoomFacade](#)

Methods

Public methods:

- [ContrastsDEqMSVoomFacade\\$new\(\)](#)
- [ContrastsDEqMSVoomFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsDEqMSVoomFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  span = 0.5,
  plot = FALSE,
  ...
)
```

Arguments:

`lfqdata` LFQData object
`modelstr` model formula string (e.g. "~ group_")
`contrasts` named character vector of contrasts
`span` lowess smoother span for vooma trend (default 0.5)

plot logical; if TRUE, plot the mean-variance trend
 ... passed to [strategy_limma](#) (e.g. trend, robust)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsDEqMSVoomFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsDEqMSVoomFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
```

ContrastsFacadeBase	<i>Base class for contrast analysis facades</i>
---------------------	---

Description

Base class for contrast analysis facades

Base class for contrast analysis facades

Details

Holds the fields and delegating methods shared by every facade in `ContrastsFacades.R` and `ContrastsChildToParentFacades.R`. Subclasses implement the unique pipeline wiring in `initialize()`: they call the private `.setup()` (input-shape check, `self$.lfqdata`, `self$.contrast_names`, `self$facade_name`), then build the model and the inner contrast object. `get_config()` defaults to the inner contrast object's configuration. The limma family additionally sets `self$.drop_na_diff <- TRUE` to drop rows whose fold change could not be estimated.

`get_contrasts()` delegates to the inner contrast object and then stamps the facade key into `modelName` via `.stamp_facade_identity()`; rescue/imputation state is carried in the `estimate_type` column set by the inner contrast classes.

Value

An R6 class generator.

Super class

`prolfqua::ContrastsInterface` -> `ContrastsFacadeBase`

Public fields

`model` fitted model object

`contrast` inner contrast object the facade delegates to

`.lfqdata` stored reference to the input `LFQData` (used by `get_missing`)

`.contrast_names` names of the requested contrasts

`facade_name` registry key stamped into the `modelName` column

`.drop_na_diff` drop rows with NA diff (limma family); default `FALSE`

Methods

Public methods:

- `ContrastsFacadeBase$get_config()`
- `ContrastsFacadeBase$get_contrasts()`
- `ContrastsFacadeBase$get_missing()`
- `ContrastsFacadeBase$get_Plotter()`
- `ContrastsFacadeBase$to_wide()`
- `ContrastsFacadeBase$clone()`

Method `get_config()`: get the `ContrastConfiguration`; defaults to the inner contrast object's configuration

Usage:

```
ContrastsFacadeBase$get_config()
```

Method `get_contrasts()`: get contrast results, stamped with the facade key

Usage:

```
ContrastsFacadeBase$get_contrasts(...)
```

Arguments:

... passed to the inner contrast object's `get_contrasts`

Method `get_missing()`: get subject x contrast pairs that could not be estimated

Usage:

```
ContrastsFacadeBase$get_missing()
```

Method `get_Plotter()`: get [ContrastsPlotter](#)

Usage:

```
ContrastsFacadeBase$get_Plotter(...)
```

Arguments:

... passed to the inner contrast object's `get_Plotter`

Method `to_wide()`: convert results to wide format

Usage:

```
ContrastsFacadeBase$to_wide(...)
```

Arguments:

... passed to the inner contrast object's `to_wide`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsFacadeBase$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
facade <- ContrastsFacadeBase$new()
inherits(facade, "ContrastsFacadeBase")
```

ContrastsFirth

Estimate contrasts using Wald Test

Description

Estimate contrasts using Wald Test

Estimate contrasts using Wald Test

Value

An R6 class generator.

Super class

`prolfqua::ContrastsInterface` -> `ContrastsFirth`

Public fields

`models` `Model`

`contrasts` character with contrasts

`model_name` name of model

`subject_id` name of column containing e.g., protein Id's

`p.adjust` function to adjust p-values (default `prolfqua::adjust_p_values`)

`contrast_result` data frame containing results of contrast computation

Methods

Public methods:

- `ContrastsFirth$new()`
- `ContrastsFirth$get_contrast_sides()`
- `ContrastsFirth$get_linfct()`
- `ContrastsFirth$get_contrasts()`
- `ContrastsFirth$clone()`

Method `new()`: initialize create Contrast

Usage:

```
ContrastsFirth$new(
  model,
  contrasts,
  p.adjust = prolfqua::adjust_p_values,
  model_name = "firth"
)
```

Arguments:

`model` a dataframe with a structure similar to that generated by `build_model`

`contrasts` a character vector with contrast specification

`p.adjust` function to adjust the p-values

`model_name` name of contrast method, default `firth`

Method `get_contrast_sides()`: get both sides of contrasts

Usage:

```
ContrastsFirth$get_contrast_sides()
```

Method `get_linfct()`: get linear functions from contrasts

Usage:

```
ContrastsFirth$get_linfct(avg = TRUE)
```

Arguments:

avg logical TRUE - get also linfct for averages

Returns: a list with the linfct-annotated model containers (models1 / models2); the shared input model is left untouched.

Method `get_contrasts()`: get table with contrast estimates

Usage:

```
ContrastsFirth$get_contrasts(all = FALSE)
```

Arguments:

all should all columns be returned (default FALSE)

Returns: data.frame with contrasts

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsFirth$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```

modi <- sim_build_models_logistf(model = "parallel3", weight_missing = 1)
contrasts <- c(Avs = "group_A - group_B", AvsCtrl = "group_A - group_Ctrl")

ctr <- ContrastsFirth$new(modi, contrasts)
ctr$get_contrast_sides()
ctr$get_linfct()
ctr$get_contrasts()

mod3 <- sim_build_models_logistf(model = "parallel3", weight_missing = 1, peptide=TRUE)

ctrpep <- ContrastsFirth$new(mod3, contrasts)
ctrpep$get_contrast_sides()

ctrpep$get_linfct()
ctrpep$get_contrasts()
pl <- ctrpep$get_Plotter()

```

ContrastsFirthFacade *Firth logistic missingness contrast analysis facade*

Description

Firth logistic missingness contrast analysis facade

Firth logistic missingness contrast analysis facade

Details

Encapsulates the pipeline: encode missingness -> [build_model_glm_protein](#) -> [ContrastsFirth](#).

Requires aggregated (protein-level) LFQData. For nested peptide-level input use [ContrastsFirthNestedFacade](#).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsFirthFacade](#)

Methods**Public methods:**

- [ContrastsFirthFacade\\$new\(\)](#)
- [ContrastsFirthFacade\\$clone\(\)](#)

Method [new\(\)](#): initialize

Usage:

[ContrastsFirthFacade\\$new](#)(lfqdata, modelstr, contrasts, ...)

Arguments:

lfqdata LFQData object
 modelstr model formula string (e.g. "~ group_")
 contrasts named character vector of contrasts
 ... ignored; accepted for dispatch compatibility with other facades

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ContrastsFirthFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 20, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsFirthFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsFirthNestedFacade

Firth logistic missingness contrast analysis facade for nested input

Description

Firth logistic missingness contrast analysis facade for nested input

Firth logistic missingness contrast analysis facade for nested input

Details

Encapsulates the pipeline: encode missingness -> [build_model_glm_peptide](#) -> [ContrastsFirth](#).

Takes nested (peptide-level) LFQData and returns protein-level fold-change estimates. For protein-level (aggregated) input use [ContrastsFirthFacade](#) instead.

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsFirthNestedFacade](#)

Methods

Public methods:

- [ContrastsFirthNestedFacade\\$new\(\)](#)
- [ContrastsFirthNestedFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsFirthNestedFacade$new(lfqdata, modelstr, contrasts, ...)
```

Arguments:

`lfqdata` nested LFQData (peptide-level)

`modelstr` model formula string (e.g. "~ group_")

`contrasts` named character vector of contrasts

`...` ignored; accepted for dispatch compatibility with other facades

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsFirthNestedFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_2\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#),

```
impute_refit_singular(), linfoct_from_model(), linfoct_matrix_contrasts(), list_facades(),
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(),
register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(),
sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(),
strategy_limpa(), strategy_logistf(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()
```

Examples

```
istar <- sim_lfq_data_peptide_config(Nprot = 20, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsFirthNestedFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsInterface	<i>Base class for all Contrasts classes</i>
--------------------	---

Description

Base class for all Contrasts classes

Base class for all Contrasts classes

Value

An R6 class generator.

Public fields

config [ContrastConfiguration](#) describing the backend's column-role mapping and behaviour flags. Subclasses should populate this in `initialize()` so generic consumers can resolve column names without hard-coding them.

Methods

Public methods:

- [ContrastsInterface\\$get_config\(\)](#)
- [ContrastsInterface\\$get_contrast_sides\(\)](#)
- [ContrastsInterface\\$get_contrasts\(\)](#)
- [ContrastsInterface\\$get_Plotter\(\)](#)
- [ContrastsInterface\\$to_wide\(\)](#)
- [ContrastsInterface\\$get_missing\(\)](#)
- [ContrastsInterface\\$get_rank\(\)](#)
- [ContrastsInterface\\$get_ora\(\)](#)
- [ContrastsInterface\\$filter_significant\(\)](#)
- [ContrastsInterface\\$contrast_summary_table\(\)](#)
- [ContrastsInterface\\$extra_artifacts\(\)](#)
- [ContrastsInterface\\$column_description\(\)](#)
- [ContrastsInterface\\$clone\(\)](#)

Method `get_config()`: Return the [ContrastConfiguration](#), constructing an LM-flavoured default on demand if a subclass has not set one.

Usage:

```
ContrastsInterface$get_config()
```

Method `get_contrast_sides()`: get table with sides of the contrast

Usage:

```
ContrastsInterface$get_contrast_sides()
```

Method `get_contrasts()`: get table with contrast results (similar to limma `topTable` function)

Usage:

```
ContrastsInterface$get_contrasts()
```

Method `get_Plotter()`: return [ContrastsPlotter](#) for the contrast results

Usage:

```
ContrastsInterface$get_Plotter(fc_threshold = 1, fdr_threshold = 0.1)
```

Arguments:

`fc_threshold` fold change threshold to show in plots

`fdr_threshold` FDR threshold to show in plots

Returns: [ContrastsPlotter](#)

Method `to_wide()`: convert contrast results to wide format

Usage:

```
ContrastsInterface$to_wide(columns = c("p.value", "FDR", "statistic"))
```

Arguments:

`columns` value columns to spread next to diff

Returns: data.frame

Method `get_missing()`: get protein × contrast pairs that could not be estimated. Returns a data.frame with hierarchy columns and contrast, or 0 rows.

Usage:

```
ContrastsInterface$get_missing()
```

Method `get_rank()`: get rank-list input table for enrichment tools.

If the backend's [ContrastConfiguration](#) has a p-value column the default rank score is signed $\text{sign}(\text{effect}) \times -\log_{10}(\text{p.value})$. Otherwise the default is the effect column itself.

Usage:

```
ContrastsInterface$get_rank(score = NULL)
```

Arguments:

`score` column name to use as rank score, or NULL to pick from the config

Returns: data.frame with subject id, contrast, and score columns.

Method `get_ora()`: get ORA input table — features passing the FDR and absolute effect-size threshold in the requested direction.

Usage:

```
ContrastsInterface$get_ora(up = TRUE, FDR_threshold = 0.05, diff_threshold = 1)
```

Arguments:

up if TRUE return positive effects, otherwise negative effects
 FDR_threshold false discovery rate threshold
 diff_threshold absolute effect-size threshold

Returns: filtered contrast data.frame for ORA input.

Method filter_significant(): Filter contrast rows that pass the FDR and effect-size threshold.

By default this is symmetric on the effect column ($|\text{effect}| > \text{diff_threshold}$). For backends whose [ContrastConfiguration](#) has `significance_directional = TRUE` (e.g. SAINTexpress, where only positive enrichment is biologically meaningful), the filter is one-sided ($\text{effect} > \text{diff_threshold}$).

Usage:

```
ContrastsInterface$filter_significant(FDR_threshold = 0.05, diff_threshold = 1)
```

Arguments:

FDR_threshold false discovery rate threshold
 diff_threshold effect-size threshold

Returns: filtered contrast data.frame.

Method contrast_summary_table(): Per-subject summary table for downstream report grobs. Returns a small data.frame with canonical column names contrast, effect, score, fdr regardless of backend.

Usage:

```
ContrastsInterface$contrast_summary_table(rounded = TRUE)
```

Arguments:

rounded if TRUE round numeric columns to 3 significant digits (matches the prolfquapp report convention)

Method extra_artifacts(): Backend-specific extra artifacts that a report should carry along with the contrast results (e.g. SAINT input tables). Default returns an empty named list.

Usage:

```
ContrastsInterface$extra_artifacts()
```

Method column_description(): column description

Usage:

```
ContrastsInterface$column_description()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ContrastsInterface$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```

int <- ContrastsInterface$new()
testthat::expect_error(int$get_contrast_sides())
testthat::expect_error(int$get_contrasts())
testthat::expect_error(int$get_missing())
# get_Plotter / to_wide / get_rank / get_ora / filter_significant call get_contrasts()
# internally, which is not implemented on the bare interface, so they surface that error.
testthat::expect_error(int$get_Plotter())
testthat::expect_error(int$to_wide())
testthat::expect_error(int$get_rank())
testthat::expect_error(int$get_ora())
testthat::expect_error(int$filter_significant())
identical(int$extra_artifacts(), list())
int$column_description()

```

ContrastsLimma

Limma-based contrasts (direct limma pipeline)

Description

Limma-based contrasts (direct limma pipeline)

Limma-based contrasts (direct limma pipeline)

Details

Uses limma's `contrasts.fit` + eBayes pipeline directly, rather than fitting per-protein lm models and then moderating. Inherits from [ContrastsInterface](#) with the same API as [Contrasts](#).

Value

An R6 class generator.

Super class

[prolfqua::ContrastsInterface](#) -> ContrastsLimma

Public fields

`model` ModelLimma object

`contrasts` named character vector of contrasts

`model_name` model name

`subject_id` columns with `subject_id` (proteinID)

`p.adjust` function to adjust p-values

`contrast_result` cached contrast results

`eBayes` logical, apply limma eBayes moderation (default TRUE). Set to FALSE to return raw/unmoderated statistics, e.g. for downstream DEqMS moderation via [ContrastsModeratedDEqMS](#).

Methods

Public methods:

- [ContrastsLimma\\$new\(\)](#)
- [ContrastsLimma\\$get_contrast_sides\(\)](#)
- [ContrastsLimma\\$get_linfct\(\)](#)
- [ContrastsLimma\\$get_contrasts\(\)](#)
- [ContrastsLimma\\$clone\(\)](#)

Method `new()`: initialize `ContrastsLimma`

Usage:

```
ContrastsLimma$new(
  model,
  contrasts,
  p.adjust = prolfqua::adjust_p_values,
  model_name = NULL,
  eBayes = TRUE
)
```

Arguments:

`model` a [ModelLimma](#) object

`contrasts` named character vector of contrasts

`p.adjust` function to adjust p-values

`model_name` name of the contrast method

`eBayes` logical, apply limma eBayes moderation (default TRUE). Set to FALSE to return raw/unmoderated statistics suitable for wrapping with [ContrastsModeratedDEqMS](#).

Method `get_contrast_sides()`: get both sides of contrasts

Usage:

```
ContrastsLimma$get_contrast_sides()
```

Method `get_linfct()`: get linear functions from contrasts

Usage:

```
ContrastsLimma$get_linfct(global = TRUE, avg = TRUE)
```

Arguments:

`global` ignored (for API compatibility)

`avg` logical, also compute avgAbd linfct

Method `get_contrasts()`: get table with contrast estimates via limma pipeline

Usage:

```
ContrastsLimma$get_contrasts(all = FALSE)
```

Arguments:

`all` should all columns be returned (default FALSE)

Returns: data.frame with contrasts

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimma$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50)
lProt <- LFQData$new(istar$data, istar$config)
lProt$rename_response("transformedIntensity")

strat <- strategy_limma("transformedIntensity ~ group_")
mod_limma <- build_model_limma(lProt, strat)

Contr <- c("dil.b_vs_a" = "group_A - group_Ctrl")
contr_limma <- ContrastsLimma$new(mod_limma, Contr)
res <- contr_limma$get_contrasts()
head(res)
stopifnot(all(c("diff", "FDR", "p.value", "statistic") %in% colnames(res)))

# Compare with prolfqua's own pipeline
modelFunction <- strategy_lm("transformedIntensity ~ group_")
mod <- build_model(lProt, modelFunction)
contr_prolfqua <- Contrasts$new(mod, Contr)
res_prolfqua <- contr_prolfqua$get_contrasts()

# fold changes should be very similar
merged <- dplyr::inner_join(
  dplyr::select(res, protein_Id, diff_limma = diff),
  dplyr::select(res_prolfqua, protein_Id, diff_prolfqua = diff),
  by = "protein_Id")
stopifnot(cor(merged$diff_limma, merged$diff_prolfqua, use = "complete.obs") > 0.99)

# Plotter works
pl <- contr_limma$get_Plotter()

# to_wide works
wide <- contr_limma$to_wide()
```

```

head(wide)

# merge_contrasts_results works
Contr2 <- c("dil.b_vs_a" = "group_A - group_Ctrl")
imp <- ContrastsLMImputeFacade$new(lProt, "~ group_", Contr2)$contrast
merged_res <- merge_contrasts_results(contr_limma, imp)

```

ContrastsLimmaFacade *Limma contrast analysis facade*

Description

Limma contrast analysis facade

Limma contrast analysis facade

Details

Encapsulates the pipeline: [strategy_limma](#) -> [build_model_limma](#) -> [ContrastsLimma](#).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsLimmaFacade](#)

Methods

Public methods:

- [ContrastsLimmaFacade\\$new\(\)](#)
- [ContrastsLimmaFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```

ContrastsLimmaFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  weights = lfqdata$nr_children_col(),
  ...
)

```

Arguments:

`lfqdata` LFQData object

`modelstr` model formula string (e.g. "~ group_")

`contrasts` named character vector of contrasts

`weights` column name for per-observation weights (default: `lfqdata$nr_children_col()`).

Pass NULL for unweighted.

... passed to [strategy_limma](#) (e.g. trend, robust)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimmaFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLimmaFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsLimmaImputeFacade

Limma contrast analysis with LOD imputation facade

Description

Limma contrast analysis with LOD imputation facade

Limma contrast analysis with LOD imputation facade

Details

Encapsulates the pipeline: `strategy_limma -> build_model_limma_impute -> ContrastsLimma`.

Proteins whose limma fit produces NA coefficients (typically from entire missing groups) are recovered by imputing missing values with the limit of detection (LOD) and refitting. The variance is borrowed from successful fits and degrees of freedom are corrected so that inference is not artificially precise from the constant imputation.

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface -> prolfqua::ContrastsFacadeBase -> ContrastsLimmaImputeFacade`

Methods

Public methods:

- `ContrastsLimmaImputeFacade$new()`
- `ContrastsLimmaImputeFacade$clone()`

Method `new()`: initialize

Usage:

```
ContrastsLimmaImputeFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  lod = NULL,
  df_method = c("observed", "borrowed"),
  weights = lfqdata$nr_children_col(),
  ...
)
```

Arguments:

`lfqdata` LFQData object (aggregated to protein level)

`modelstr` model formula string (e.g. "~ group_")

`contrasts` named character vector of contrasts

`lod` numeric limit of detection; if NULL, auto-computed from data

`df_method` "observed" uses $\max(n_{\text{observed}} - p, 1)$; "borrowed" uses median df from successful fits

`weights` column name for per-observation weights (default: `lfqdata$nr_children_col()`). Pass NULL for unweighted.

... passed to `strategy_limma` (e.g. trend, robust)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimmaImputeFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaVoom](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 30, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLimmaImputeFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsLimmaVoomFacade

Limma-voom contrast analysis facade

Description

Limma-voom contrast analysis facade

Limma-voom contrast analysis facade

Details

Encapsulates the pipeline: [strategy_limma](#) -> [build_model_limma_voom](#) -> [ContrastsLimma](#).

Uses vooma-style precision weights derived from a mean-variance trend, optionally combined with external weights (e.g. peptide/precursor counts).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsLimmaVoomFacade](#)

Methods**Public methods:**

- [ContrastsLimmaVoomFacade\\$new\(\)](#)
- [ContrastsLimmaVoomFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsLimmaVoomFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  weights = lfqdata$nr_children_col(),
  span = 0.5,
  plot = FALSE,
  ...
)
```

Arguments:

`lfqdata` LFQData object
`modelstr` model formula string (e.g. "~ group_")
`contrasts` named character vector of contrasts
`weights` column name for per-observation weights (default: `lfqdata$nr_children_col()`).
 Pass NULL for unweighted.
`span` lowess smoother span for vooma trend (default 0.5)
`plot` logical; if TRUE, plot the mean-variance trend
`...` passed to [strategy_limma](#) (e.g. trend, robust)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimmaVoomFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#),

```

build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance_
compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(),
get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(),
impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(),
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(),
register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(),
sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(),
strategy_limpa(), strategy_logistf(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

```

Examples

```

istar <- sim_lfq_data_protein_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLimmaVoomFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()

```

ContrastsLimmaVoomImputeFacade

Limma-voom contrast analysis with LOD imputation facade

Description

Limma-voom contrast analysis with LOD imputation facade

Limma-voom contrast analysis with LOD imputation facade

Details

Encapsulates the pipeline: `strategy_limma -> build_model_limma_voom_impute -> ContrastsLimma`.

Combines vooma precision weights with LOD imputation for proteins whose fit produces NA coefficients (typically from entire missing groups).

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface -> prolfqua::ContrastsFacadeBase -> ContrastsLimmaVoomImputeFacade`

Methods

Public methods:

- `ContrastsLimmaVoomImputeFacade$new()`
- `ContrastsLimmaVoomImputeFacade$clone()`

Method `new()`: initialize

Usage:

```

ContrastsLimmaVoomImputeFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  lod = NULL,
  df_method = c("observed", "borrowed"),
  weights = lfqdata$nr_children_col(),
  span = 0.5,
  plot = FALSE,
  ...
)

```

Arguments:

lfqdata LFQData object (aggregated to protein level)

modelstr model formula string (e.g. "~ group_")

contrasts named character vector of contrasts

lod numeric limit of detection; if NULL, auto-computed from data

df_method "observed" uses $\max(n_{\text{observed}} - p, 1)$; "borrowed" uses median df from successful fits

weights column name for per-observation weights (default: `lfqdata$nr_children_col()`). Pass NULL for unweighted.

span lowess smoother span for vooma trend (default 0.5)

plot logical; if TRUE, plot the mean-variance trend

... passed to [strategy_limma](#) (e.g. trend, robust)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimmaVoomImputeFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```

istar <- sim_lfq_data_protein_config(Nprot = 30, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLimmaVoomImputeFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()

```

ContrastsLimpaFacade *Limpa contrast analysis facade*

Description

Limpa contrast analysis facade

Limpa contrast analysis facade

Details

Encapsulates the pipeline: [strategy_limpa](#) -> [build_model_limpa](#) -> [ContrastsLimma](#).

Operates as a "same"-level facade: it consumes whatever aggregated LFQData prolfqua's normal aggregation pipeline produced and fits limpa at that hierarchy level. If `config$opt_se` is set (e.g. when the input came from [AggregateLimpa](#)), the per-observation standard error is used as a vooma precision-weight predictor; otherwise plain vooma is fit. The `config$nr_children` column is required and is used to flag imputed observations (`nr_children == 0`) for vooma's imputation-aware DF correction.

For nested (peptide/precursor) input that should be rolled up to proteins via limpa's DPC quantification, use [ContrastsLimpaNestedFacade](#) instead — that facade owns the [AggregateLimpa](#) pre-step.

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsLimpaFacade](#)

Methods**Public methods:**

- [ContrastsLimpaFacade\\$new\(\)](#)
- [ContrastsLimpaFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsLimpaFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  plot = FALSE,
  span = NULL,
  ...
)
```

Arguments:

lfqdata aggregated LFQData. If `config$opt_se` is set (e.g. from [AggregateLimpa](#)), the SE column is used as a vooma precision-weight predictor; otherwise vooma is fit without an external predictor.

modelstr model formula string (e.g. "`~ group_`")

contrasts named character vector of contrasts

plot logical; if TRUE, plot the vooma mean-variance trend

span lowess smoother span (NULL = auto)

... passed to [strategy_limpa](#) (e.g. trend, robust)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimpaFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_2\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()
lfqdata <- LFQData$new(istar$data, istar$config)
```

```

lfqdata <- lfqdata$get_Transformer()$log2()$lfq
agg <- AggregateLimpa$new(lfqdata, "protein")
lfq_agg <- agg$aggregate()
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLimpaFacade$new(lfq_agg, "~ group_", contrasts)
head(fa$get_contrasts())

```

ContrastsLimpaNestedFacade

Limpa contrast analysis facade for nested input

Description

Limpa contrast analysis facade for nested input

Limpa contrast analysis facade for nested input

Details

Encapsulates the full precursor -> protein pipeline: [AggregateLimpa](#) -> [strategy_limpa](#) -> [build_model_limpa](#) -> [ContrastsLimma](#).

Takes nested (precursor/peptide-level) LFQData, runs limpa's DPC-based aggregation internally to produce protein-level expression with posterior standard errors, then fits a vooma model with imputation-aware precision weights.

For protein-level input that was already aggregated upstream via [AggregateLimpa](#), use [ContrastsLimpaFacade](#) instead.

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsLimpaNestedFacade](#)

Methods

Public methods:

- [ContrastsLimpaNestedFacade\\$new\(\)](#)
- [ContrastsLimpaNestedFacade\\$clone\(\)](#)

Method new(): initialize

Usage:

```

ContrastsLimpaNestedFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  prefix = "protein",
  dpc_slope = 0.8,
  plot = FALSE,
  span = NULL,
  ...
)

```

Arguments:

lfqdata nested LFQData (precursor/peptide-level, log2-transformed)
 modelstr model formula string (e.g. "~ group_")
 contrasts named character vector of contrasts
 prefix prefix for the aggregated hierarchy level (default "protein")
 dpc_slope DPC slope parameter passed to [AggregateLimpa](#) (default 0.8)
 plot logical; if TRUE, plot the vooma mean-variance trend
 span lowess smoother span (NULL = auto)
 ... passed to [strategy_limpa](#) (e.g. trend, robust)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ContrastsLimpaNestedFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_residues\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
if (requireNamespace("limpa", quietly = TRUE)) {
  istar <- prolfqua::sim_lfq_data_peptide_config(Nprot = 10)
  lfqdata <- LFQData$new(istar$data, istar$config)
  lfqdata <- lfqdata$get_Transformer()$log2()$lfq
  contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
  fa <- ContrastsLimpaNestedFacade$new(lfqdata, "~ group_", contrasts)
  head(fa$get_contrasts())
}
```

ContrastsLmerNestedFacade

Lmer (mixed-model) contrast analysis facade for nested input

Description

Lmer (mixed-model) contrast analysis facade for nested input

Lmer (mixed-model) contrast analysis facade for nested input

Details

Encapsulates the pipeline: `strategy_lmer` -> `build_model` -> `Contrasts` -> `ContrastsModerated`.

Takes child-hierarchy LFQData (e.g. peptide-level measurements nested within proteins) and returns protein-level fold-change estimates.

modelstr may be either a fixed-effects-only formula (e.g. "`~ group_`") - in which case the facade appends `(1 | <deepest child hierarchy key>) + (1 | <sample name>)` derived from the LFQData - or a full mixed-model formula already containing random effects, which is passed through unchanged.

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface` -> `prolfqua::ContrastsFacadeBase` -> `ContrastsLmerNestedFacade`

Methods

Public methods:

- `ContrastsLmerNestedFacade$new()`
- `ContrastsLmerNestedFacade$clone()`

Method `new()`: initialize

Usage:

```
ContrastsLmerNestedFacade$new(lfqdata, modelstr, contrasts, ...)
```

Arguments:

lfqdata nested LFQData (subject_id strict subset of hierarchy_keys)

modelstr model formula RHS as string. Fixed effects only (e.g. "`~ group_`") - random effects are derived from lfqdata as `(1 | <deepest child key>) + (1 | <sample name>)`. A formula already containing random-effect bars is passed through unchanged.

contrasts named character vector of contrasts

... passed to `strategy_lmer`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLmerNestedFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNeste](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata <- lfqdata$get_Transformer()$log2()$lfq
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
# Random effects derived from lfqdata; equivalent to passing the full
# mixed-model formula explicitly.
fa <- ContrastsLmerNestedFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsLMFacade	<i>LM contrast analysis facade</i>
-------------------	------------------------------------

Description

LM contrast analysis facade
LM contrast analysis facade

Details

Encapsulates the pipeline: [strategy_lm](#) -> [build_model](#) -> [Contrasts](#) -> [ContrastsModerated](#).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsLMFacade](#)

Methods

Public methods:

- [ContrastsLMFacade\\$new\(\)](#)
- [ContrastsLMFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsLMFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  weights = lfqdata$nr_children_col(),
  ...
)
```

Arguments:

`lfqdata` LFQData object

`modelstr` model formula string (e.g. "~ group_")

`contrasts` named character vector of contrasts

`weights` column name for per-observation weights (default: `lfqdata$nr_children_col()`).

Pass NULL for unweighted.

... passed to [strategy_lm](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsLMFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#),

```
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(),
register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(),
sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(),
strategy_limpa(), strategy_logistf(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()
```

Examples

```
istar <- sim_lfq_data_protein_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLMFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsLMImputeFacade

LM contrast analysis with LOD imputation facade

Description

LM contrast analysis with LOD imputation facade

LM contrast analysis with LOD imputation facade

Details

Encapsulates the pipeline: `strategy_lm` -> `build_model` (with `impute = TRUE`) -> `Contrasts` -> `ContrastsModerated`.

Proteins whose initial lm fit fails or produces NA coefficients are re-fitted after imputing missing values with the limit of detection (LOD). The covariance matrix is borrowed from successful fits so that the variance is not underestimated by the constant imputation.

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface` -> `prolfqua::ContrastsFacadeBase` -> `ContrastsLMImputeFacade`

Methods

Public methods:

- `ContrastsLMImputeFacade$new()`
- `ContrastsLMImputeFacade$clone()`

Method `new()`: initialize

Usage:

```

ContrastsLMImputeFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  lod = NULL,
  borrow_method = c("sigma", "vcov"),
  df_method = c("observed", "borrowed"),
  weights = lfqdata$nr_children_col(),
  ...
)

```

Arguments:

lfqdata LFQData object (aggregated to protein level)
 modelstr model formula string (e.g. "~ group_")
 contrasts named character vector of contrasts
 lod numeric limit of detection; if NULL, auto-computed from data
 borrow_method "sigma" borrows scalar sigma and uses per-protein $(X'X)^{-1}$; "vcov" borrows element-wise median of full vcov matrices
 df_method "observed" uses $\max(n_{\text{observed}} - p, 1)$; "borrowed" uses median df from successful fits
 weights column name for per-observation weights (default: lfqdata\$nr_children_col()). Pass NULL for unweighted.
 ... passed to [strategy_lm](#)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ContrastsLMImputeFacade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```

istar <- sim_lfq_data_protein_config(Nprot = 30, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsLMImputeFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()

```

ContrastsModerated	<i>Limma moderated contrasts</i>
--------------------	----------------------------------

Description

Limma moderated contrasts
 Limma moderated contrasts

Value

An R6 class generator.

Super class

[prolfqua::ContrastsInterface](#) -> ContrastsModerated

Public fields

Contrast Class implementing the Contrast interface
 model_name name of model
 subject_id columns with subject_id (proteinID)
 p.adjust function to adjust p-values
 variance_floor optional lower bound for posterior variances

Methods**Public methods:**

- [ContrastsModerated\\$new\(\)](#)
- [ContrastsModerated\\$get_contrast_sides\(\)](#)
- [ContrastsModerated\\$get_linfct\(\)](#)
- [ContrastsModerated\\$get_contrasts\(\)](#)
- [ContrastsModerated\\$clone\(\)](#)

Method new(): initialize

Usage:

```

ContrastsModerated$new(
  Contrast,
  model_name = paste0(Contrast$model_name, "_moderated"),
  p.adjust = prolfqua::adjust_p_values,
  variance_floor = NULL
)

```

Arguments:

`Contrast` class implementing the `ContrastInterface`

`model_name` name of the model

`p.adjust` function to adjust p-values - default BH

`variance_floor` optional positive lower bound for posterior variances

Method `get_contrast_sides()`: get both sides of contrasts

Usage:

`ContrastsModerated$get_contrast_sides()`

Method `get_linfct()`: get linear functions from contrasts

Usage:

`ContrastsModerated$get_linfct(global = TRUE)`

Arguments:

`global` logical TRUE - get the a linear functions for all models, FALSE - linear function for each model

Method `get_contrasts()`: applies limma moderation

Usage:

`ContrastsModerated$get_contrasts(all = FALSE)`

Arguments:

`all` should all columns be returned (default FALSE)

`global` use a global linear function (determined by `get_linfct`)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`ContrastsModerated$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

[moderated_p_limma](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#).

```
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua())
```

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50)
protIntensity <- istar$data
config <- istar$config

lProt <- LFQData$new(protIntensity, config)
lProt$rename_response("transformedIntensity")
modelFunction <-
  strategy_lm("transformedIntensity ~ group_")
mod <- build_model(
  lProt,
  modelFunction)

Contr <- c("dil.b_vs_a" = "group_A - group_Ctrl")
contrast <- prolfqua::Contrasts$new(mod,
  Contr)
contrast <- ContrastsModerated$new(contrast)
bb <- contrast$get_contrasts()
# LOD-imputed refit for proteins the plain lm cannot estimate
imp <- ContrastsLMImputeFacade$new(lProt, "~ group_", Contr)$contrast

contrast$get_contrasts() |> dim()
(xx <- imp$get_contrasts()) |> dim()
merged <- merge_contrasts_results(contrast, imp)

merged$more$get_contrasts() |> dim()
stopifnot(all(dim(merged$merged$get_contrasts()) == c(50, 16)))
stopifnot(all(dim(merged$same$get_contrasts()) == c(49, 16)))

cs <- contrast$get_contrast_sides()
cslf <- contrast$get_linfct()
ctr <- contrast$get_contrasts()
ctrwide <- contrast$to_wide()
cp <- contrast$get_Plotter()

print(cp$histogram())$p.value, vp=NULL)
print(cp$histogram())$FDR, vp = NULL)

cp$volcano()
cp$ma_plot()
```

Description

DEqMS count-dependent moderated contrasts

DEqMS count-dependent moderated contrasts

Details

Decorator that wraps any Contrasts object and applies count-dependent empirical Bayes variance shrinkage. Similar to [ContrastsModerated](#) but the prior variance depends on the number of quantified peptides/PSMs per protein: proteins with many peptides get less shrinkage, proteins with few peptides get more.

Value

An R6 class generator.

Super class

[prolfqua::ContrastsInterface](#) -> ContrastsModeratedDEqMS

Public fields

Contrast Class implementing the Contrast interface
 count_df data.frame with subject_id + count column
 count_column name of the count column in count_df
 loess_span span parameter for LOESS fit
 model_name name of model
 subject_id columns with subject_id (proteinID)
 p.adjust function to adjust p-values

Methods**Public methods:**

- [ContrastsModeratedDEqMS\\$new\(\)](#)
- [ContrastsModeratedDEqMS\\$get_contrast_sides\(\)](#)
- [ContrastsModeratedDEqMS\\$get_linfct\(\)](#)
- [ContrastsModeratedDEqMS\\$get_contrasts\(\)](#)
- [ContrastsModeratedDEqMS\\$clone\(\)](#)

Method new(): initialize

Usage:

```
ContrastsModeratedDEqMS$new(
  Contrast,
  count_df,
  count_column,
  loess_span = 0.75,
  model_name = paste0(Contrast$model_name, "_DEqMS"),
  p.adjust = prolfqua::adjust_p_values
)
```

Arguments:

Contrast class implementing the ContrastInterface
 count_df data.frame with subject_id columns and a count column
 count_column name of the count column in count_df
 loess_span span for LOESS variance fit (default 0.75)
 model_name name of the model
 p.adjust function to adjust p-values - default BH

Method get_contrast_sides(): get both sides of contrasts

Usage:

```
ContrastsModeratedDEqMS$get_contrast_sides()
```

Method get_linfct(): get linear functions from contrasts

Usage:

```
ContrastsModeratedDEqMS$get_linfct(global = TRUE)
```

Arguments:

global logical TRUE - get linear functions for all models

Method get_contrasts(): applies DEqMS-style count-dependent moderation

Usage:

```
ContrastsModeratedDEqMS$get_contrasts(all = FALSE)
```

Arguments:

all should all columns be returned (default FALSE)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ContrastsModeratedDEqMS$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

[moderated_p_deqms_long](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```

istar <- sim_lfq_data_protein_config(Nprot = 50)
protIntensity <- istar$data
config <- istar$config

lProt <- LFQData$new(protIntensity, config)
lProt$rename_response("transformedIntensity")
modelFunction <-
  strategy_lm("transformedIntensity ~ group_")
mod <- build_model(
  lProt,
  modelFunction)

Contr <- c("dil.b_vs_a" = "group_A - group_Ctrl")
contrast <- prolfqua::Contrasts$new(mod, Contr)

# Build count_df from config
count_df <- dplyr::select(protIntensity,
  dplyr::all_of(c(config$hierarchy_keys_depth(), "nrPeptides"))) |>
  dplyr::distinct()

deqms <- ContrastsModeratedDEqMS$new(contrast,
  count_df = count_df,
  count_column = "nrPeptides")

bb <- deqms$get_contrasts()
stopifnot(all(c("diff", "p.value", "FDR", "sigma") %in% colnames(bb)))

# Merge with LOD-imputed refits for proteins the plain lm cannot estimate
imp <- ContrastsLMImputeFacade$new(lProt, "~ group_", Contr)$contrast
merged <- merge_contrasts_results(deqms, imp)

cs <- deqms$get_contrast_sides()
cslf <- deqms$get_linfct()
ctrwide <- deqms$to_wide()
cp <- deqms$get_Plotter()
cp$volcano()

```

ContrastsPlotter

plot contrasts

Description

plot contrasts

plot contrasts

Value

An R6 class generator.

Public fields

model_name of column with model name
 subject_id hierarchy key columns
 diff column with fold change differences
 contrast column with contrasts names, default "contrast"
 group crosstalk group name for linked brushing, default "BB"
 volcano_spec volcano plot specification
 score_spec score plot specification
 histogram_spec plot specification
 fcthresh fold change threshold
 avg.abundance name of column containing avg abundance values.

Methods**Public methods:**

- `ContrastsPlotter$new()`
- `ContrastsPlotter$histogram()`
- `ContrastsPlotter$histogram_diff()`
- `ContrastsPlotter$volcano()`
- `ContrastsPlotter$volcano_plotly()`
- `ContrastsPlotter$ma_plot()`
- `ContrastsPlotter$ma_plotly()`
- `ContrastsPlotter$score_plot()`
- `ContrastsPlotter$barplot_threshold()`
- `ContrastsPlotter$clone()`

Method `new()`: create `ContrastsPlotter`

Usage:

```

ContrastsPlotter$new(
  contrast_df,
  subject_id,
  volcano = list(list(score = "FDR", thresh = 0.1)),
  histogram = list(list(score = "p.value", xlim = c(0, 1, 0.05)), list(score = "FDR",
    xlim = c(0, 1, 0.05))),
  score = list(list(score = "statistic", thresh = NULL)),
  fcthresh = 1,
  modelName = "modelName",
  diff = "diff",
  contrast = "contrast",
  avg.abundance = "avgAbd",
  group = "BB"
)

```

Arguments:

contrast_df frame with contrast data
 subject_id columns containing subject Identifier
 volcano which score to plot and which ablines to add.

histogram which scores to plot and which range (x) should be shown.
 score score parameters
 fctthresh default 1 (log2 FC threshold)
 modelName name of the colour column. Defaults to "modelName"; when left at the default and the contrast carries an estimate_type column, that column is used for colouring instead (so observed vs imputed/fallback rows stay distinguishable).
 diff fold change (difference) diff column
 contrast contrast column
 avg.abundance name of column with average abundance
 group crosstalk group name for linked brushing, default "BB"

Method histogram(): plot histogram of selected scores (e.g. p-value, FDR, t-statistics)

Usage:

```
ContrastsPlotter$histogram()
```

Method histogram_diff(): plot histogram of differences (diff) fold change

Usage:

```
ContrastsPlotter$histogram_diff(binwidth = 0.05)
```

Arguments:

binwidth width of bin in histogram

Method volcano(): volcano plots (fold change vs FDR)

Usage:

```
ContrastsPlotter$volcano(
  colour = self$model_name,
  legend = TRUE,
  scales = c("fixed", "free", "free_x", "free_y"),
  min_score = NULL
)
```

Arguments:

colour column name with color information default modelName

legend default TRUE

scales default fixed [facet_wrap](#), scales argument

min_score optional lower bound for p-values or FDR values. If NULL, only exact zero and negative values are replaced with the smallest positive observed value in the same score column.

Method volcano_plotly(): plotly volcano plots

Usage:

```
ContrastsPlotter$volcano_plotly(
  colour = self$model_name,
  legend = TRUE,
  scales = c("fixed", "free", "free_x", "free_y"),
  min_score = NULL
)
```

Arguments:

colour column in contrast matrix with colour coding

legend default TRUE

scales default fixed `facet_wrap`, scales argument
 min_score optional lower bound for p-values or FDR values. If NULL, only exact zero and negative values are replaced with the smallest positive observed value in the same score column.

Returns: list of ggplots

Method `ma_plot()`: ma plot

MA plot displays the effect size estimate as a function of the mean protein intensity across groups. Each dot represents an observed protein. Red horizontal lines represent the fold-change threshold. Sometimes measured effects sizes (differences between samples groups) are biased by the signal intensity (here protein abundance). Such systematic effects can be explored using MA-plots.

Usage:

```
ContrastsPlotter$ma_plot(
  fc = self$fcthresh,
  colour = self$model_name,
  legend = TRUE,
  rank = TRUE
)
```

Arguments:

fc fold change abline
 colour column in contrast matrix with colour coding
 legend enable legend default TRUE
 rank default FALSE, if TRUE then rank of avgAbd is used.

Returns: ggplot

Method `ma_plotly()`: ma plotly

Usage:

```
ContrastsPlotter$ma_plotly(
  fc = self$fcthresh,
  colour = self$model_name,
  legend = TRUE,
  rank = FALSE
)
```

Arguments:

fc horizontal lines
 colour column in contrast matrix with colour coding.
 legend enable legend default TRUE
 rank default FALSE, if TRUE then rank of avgAbd is used.

Returns: list of ggplots

Method `score_plot()`: plot a score against the log2 fc e.g. t-statistic

Usage:

```
ContrastsPlotter$score_plot(scorespec, colour = self$model_name, legend = TRUE)
```

Arguments:

scorespec list(score="statistics", fcthres = 2, thresh = 5)
 colour column with colour coding
 legend enable legend default TRUE

Returns: list of ggplots

Method `barplot_threshold()`: shows the number of significant proteins per contrasts

Usage:

```
ContrastsPlotter$barplot_threshold()
```

Returns: list which contains ggplots and summary tables

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsPlotter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Other plotting: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config(Nprot = 20)
model_name <- "Model"
modelFunction <-
  strategy_lmer("abundance ~ group_ + (1 | peptide_Id) + (1|sample)",
    model_name = model_name)
pepIntensity <- istar$data
config <- istar$config

mod <- build_model(
```

```

    pepIntensity,
    modelFunction,
    model_name = model_name,
    subject_id = config$hierarchy_keys_depth())

  Contr <- c("group_A_vs_Ctrl" = "group_A - group_Ctrl",
    "group_B_vs_Ctrl" = "group_B - group_Ctrl"
  )
  contrast <- prolfqua::Contrasts$new(mod,
    Contr)
  contr <- contrast$get_contrasts()
  cp <- ContrastsPlotter$new(contr,
    contrast$subject_id,
    volcano = list(list(score = "FDR", thresh = 0.1)),
    histogram = list(list(score = "p.value", xlim = c(0,1,0.05)),
      list(score = "FDR", xlim = c(0,1,0.05))),
    score = list(list(score = "statistic", thresh = 5))
  )

  stopifnot("plotly" %in% class(cp$volcano_plotly())$FDR))
  stopifnot("ggplot" %in% class(cp$score_plot(legend=FALSE)$statistic))
  p <- cp$histogram()
  stopifnot("ggplot" %in% class(p$FDR))
  stopifnot("ggplot" %in% class(p$p.value))
  res <- cp$volcano()
  stopifnot("ggplot" %in% class(res$FDR))
  res_plotly <- cp$volcano_plotly()
  stopifnot("plotly" %in% class(res_plotly$FDR))
  stopifnot("ggplot" %in% class(cp$ma_plot()))
  stopifnot("plotly" %in% class(cp$ma_plotly(rank=TRUE)))
  res <- cp$barplot_threshold()
  stopifnot("ggplot" %in% class(res$FDR$plot))
  stopifnot("ggplot" %in% class(cp$histogram_diff()))

```

ContrastsRfitFacade	<i>Rfit rank-based regression contrast analysis facade</i>
---------------------	--

Description

Rfit rank-based regression contrast analysis facade

Rfit rank-based regression contrast analysis facade

Details

Encapsulates the pipeline: [strategy_rfit](#) -> [build_model](#) -> [Contrasts](#) -> [ContrastsModerated](#), using [rfit](#) rank-based linear regression. Output is the standard Wald contrast schema; the empirical-Bayes moderation shrinks the rank-based scale across proteins.

Unlike [ContrastsLMFacade](#) this backend takes no observation weights (rfit has no weights argument), so nr_children weighting is not applied.

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface -> prolfqua::ContrastsFacadeBase -> ContrastsRfitFacade`

Methods

Public methods:

- `ContrastsRfitFacade$new()`
- `ContrastsRfitFacade$clone()`

Method `new()`: initialize

Usage:

```
ContrastsRfitFacade$new(lfqdata, modelstr, contrasts, ...)
```

Arguments:

`lfqdata` LFQData object
`modelstr` model formula string (e.g. "~ group_")
`contrasts` named character vector of contrasts
`...` passed to `strategy_rfit`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsRfitFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsRfitFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsRfitImputeFacade

Rfit rank-regression contrast analysis with LOD imputation facade

Description

Rfit rank-regression contrast analysis with LOD imputation facade

Rfit rank-regression contrast analysis with LOD imputation facade

Details

The rank-based analogue of [ContrastsLMImputeFacade](#). Encapsulates the pipeline: [strategy_rfit](#) -> [build_model_impute](#) -> [Contrasts](#) -> [ContrastsModerated](#), using [rfit](#).

Proteins whose initial [rfit](#) fit fails, is singular, or has an incomplete design are re-fitted after imputing missing values with the limit of detection (LOD). Uncertainty is borrowed from successful donor fits so it is not underestimated by the constant imputation.

Differences from [ContrastsLMImputeFacade](#):

- Covariance borrowing is **vcov-only** (element-wise median of the donors' full named covariance matrices). [rfit](#) exposes no `lm`-style `cov.unscaled`, so the scalar-sigma borrowing mode is not available and is not exposed.
- Borrowing is **fail-hard** (`on_misalign = "fail"`): if the donor covariance matrices cannot be aligned by coefficient name, the rescue is skipped and those proteins remain in `get_missing()` rather than carrying an `lm`-specific approximation.
- No observation weights ([rfit](#) has no `weights` argument), matching plain [rfit](#).

`modelName` is the facade key `"rfit_impute"`; rescued rows are flagged in the `estimate_type` column as `"lod_imputed"`, the same convention used by [ContrastsLMImputeFacade](#).

Value

An R6 class generator.

Super classes

[prolfqua::ContrastsInterface](#) -> [prolfqua::ContrastsFacadeBase](#) -> [ContrastsRfitImputeFacade](#)

Methods

Public methods:

- [ContrastsRfitImputeFacade\\$new\(\)](#)
- [ContrastsRfitImputeFacade\\$clone\(\)](#)

Method `new()`: initialize

Usage:

```
ContrastsRfitImputeFacade$new(
  lfqdata,
  modelstr,
  contrasts,
  lod = NULL,
  df_method = c("observed", "borrowed"),
  ...
)
```

Arguments:

`lfqdata` LFQData object (aggregated to protein level)
`modelstr` model formula string (e.g. "~ group_")
`contrasts` named character vector of contrasts
`lod` numeric limit of detection; if NULL, auto-computed from data
`df_method` "observed" uses $\max(n_{\text{observed}} - p, 1)$; "borrowed" uses median df from successful fits
`...` passed to [strategy_rfit](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsRfitImputeFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[ContrastsLMImputeFacade](#), [ContrastsRfitFacade](#), [build_model_impute](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#).

```
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()
```

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 30, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsRfitImputeFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsRLMFacade	<i>RLM contrast analysis facade</i>
--------------------	-------------------------------------

Description

RLM contrast analysis facade

RLM contrast analysis facade

Details

Encapsulates the pipeline: `strategy_rlm -> build_model -> Contrasts -> ContrastsModerated`.

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface -> prolfqua::ContrastsFacadeBase -> ContrastsRLMFacade`

Methods

Public methods:

- `ContrastsRLMFacade$new()`
- `ContrastsRLMFacade$clone()`

Method `new()`: initialize

Usage:

`ContrastsRLMFacade$new(lfqdata, modelstr, contrasts, ...)`

Arguments:

`lfqdata` LFQData object

`modelstr` model formula string (e.g. "`~ group_`")

`contrasts` named character vector of contrasts

`...` passed to `strategy_rlm`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
ContrastsRLMFacade\$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsRLMFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsROPECA	<i>ROPECA reproducibility-optimization method</i>
-----------------	---

Description

ROPECA reproducibility-optimization method
ROPECA reproducibility-optimization method

Details

ROPECA optimizes the reproducibility of statistical testing on peptide-level and aggregates the peptide-level changes to determine differential protein-level expression.

Value

An R6 class generator.

Super class

`prolfqua::ContrastsInterface` -> `ContrastsROPECA`

Public fields

`Contrast` Contrast
`contrast_result` contrast result
`model_name` model name
`subject_id` columns with protein ID's
`p.adjust` method to use for p.value adjustment

Methods**Public methods:**

- `ContrastsROPECA$new()`
- `ContrastsROPECA$get_contrast_sides()`
- `ContrastsROPECA$get_linfct()`
- `ContrastsROPECA$get_contrasts()`
- `ContrastsROPECA$get_Plotter()`
- `ContrastsROPECA$to_wide()`
- `ContrastsROPECA$clone()`

Method `new()`: initialize

Usage:

```
ContrastsROPECA$new(
  Contrast,
  model_name = "ROPECA",
  p.adjust = prolfqua::adjust_p_values
)
```

Arguments:

`Contrast` e.g. instance of `Contrasts` class, or `ContrastsModerated`
`model_name` default ROPECA
`p.adjust` function to use for p.value adjustment

Method `get_contrast_sides()`: show names of contrasts

Usage:

```
ContrastsROPECA$get_contrast_sides()
```

Returns: data.frame

Method `get_linfct()`: get linear function used to determine contrasts

Usage:

```
ContrastsROPECA$get_linfct()
```

Returns: data.frame

Method `get_contrasts()`: get contrasts

Usage:

```
ContrastsROPECA$get_contrasts(all = FALSE)
```

Arguments:

`all` should all columns be returned (default FALSE)

`global` use a global linear function (determined by `get_linfct`)

Returns: data.frame

Method `get_Plotter()`: get [ContrastsPlotter](#)

Usage:

```
ContrastsROPECA$get_Plotter(fdr_threshold = 0.1, fc_threshold = 2)
```

Arguments:

`fdr_threshold` FDR threshold

`fc_threshold` FC threshold

Returns: [ContrastsPlotter](#)

Method `to_wide()`: convert to wide format

Usage:

```
ContrastsROPECA$to_wide(
  columns = c("beta.based.significance", "FDR.beta.based.significance")
)
```

Arguments:

`columns` value column default beta.based.significance

Returns: data.frame

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsROPECA$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[summary_ROPECA_median_p_scaled](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#),

```

get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(),
linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_res
model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(),
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()

```

Examples

```

istar <- prolfqua::sim_lfq_data_peptide_config(Nprot=20)
istar_data <- istar$data
modelFunction <-
  strategy_lm("abundance ~ group_")
pepIntensity <- istar_data
config <- istar$config$clone(deep = TRUE)
config$hierarchy_depth <- 2
config$hierarchy_keys_depth()

mod <- build_model(
  pepIntensity,
  modelFunction,
  subject_id = config$hierarchy_keys_depth())

Contr <- c("AvsCtrl" = "group_A - group_Ctrl")

contr <- prolfqua::Contrasts$new(mod, Contr)
dim(contr$get_contrasts())
contrM <- prolfqua::ContrastsModerated$new(contr)
dim(contrM$get_contrasts())
contrast <- prolfqua::ContrastsROPECA$new(contrM)
contrast$get_contrasts()
contrast <- prolfqua::ContrastsROPECA$new(contr)
tmp <- contrast$get_contrasts()
dim(tmp)
pl <- contrast$get_Plotter()
contrast$to_wide()
contrast$get_linfct()
contrast$get_contrast_sides()
pl$histogram()
pl$ma_plot()

```

ContrastsROPECANestedFacade

ROPECA contrast analysis facade for nested input

Description

ROPECA contrast analysis facade for nested input

ROPECA contrast analysis facade for nested input

Details

Encapsulates the pipeline: `strategy_lm` -> `build_model` -> `Contrasts` -> `ContrastsROPECA`.

ROPECA fits per-peptide linear models and aggregates peptide-level p-values to the protein level. Input must be nested LFQData (peptide subject_id strictly below the protein hierarchy).

Value

An R6 class generator.

Super classes

`prolfqua::ContrastsInterface` -> `prolfqua::ContrastsFacadeBase` -> `ContrastsROPECANestedFacade`

Methods

Public methods:

- `ContrastsROPECANestedFacade$new()`
- `ContrastsROPECANestedFacade$get_contrasts()`
- `ContrastsROPECANestedFacade$get_Plotter()`
- `ContrastsROPECANestedFacade$to_wide()`
- `ContrastsROPECANestedFacade$clone()`

Method `new()`: initialize

Usage:

```
ContrastsROPECANestedFacade$new(lfqdata, modelstr, contrasts, ...)
```

Arguments:

`lfqdata` nested LFQData (peptide-level)

`modelstr` model formula string (e.g. "`~ group_`")

`contrasts` named character vector of contrasts

`...` passed to `strategy_lm`

Method `get_contrasts()`: Get contrast results with standardized column names.

ROPECA's `beta.based.significance` is mapped to `p.value` and `FDR.beta.based.significance` to `FDR`.

Columns not directly produced by ROPECA are derived heuristically:

- `std.error` = `diff / statistic` (algebraic: `t = estimate / SE`)
- `sigma` = `mad.estimate` (MAD of peptide fold changes)
- `df` = `n_not_na` (number of contributing peptides)
- `conf.low/conf.high` via `diff +/- qt(0.975, df) * |std.error|`

Usage:

```
ContrastsROPECANestedFacade$get_contrasts()
```

Method `get_Plotter()`: get `ContrastsPlotter` (uses standardized column names)

Usage:

```
ContrastsROPECANestedFacade$get_Plotter(fc_threshold = 2, fdr_threshold = 0.1)
```

Arguments:

`fc_threshold` fold change threshold

`fdr_threshold` FDR threshold

Method `to_wide()`: convert results to wide format

Usage:

```
ContrastsROPECANestedFacade$to_wide(columns = c("p.value", "FDR", "statistic"))
```

Arguments:

`columns` value columns to pivot, default `c("p.value", "FDR", "statistic")`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsROPECANestedFacade$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata <- lfqdata$get_Transformer()$log2()$lfq
contrasts <- c("A_vs_Ctrl" = "group_A - group_Ctrl")
fa <- ContrastsROPECANestedFacade$new(lfqdata, "~ group_", contrasts)
head(fa$get_contrasts())
fa$to_wide()
```

ContrastsTable	<i>holds results when contrasts are added.</i>
----------------	--

Description

holds results when contrasts are added.

holds results when contrasts are added.

Value

An R6 class generator.

Super class

`prolfqua::ContrastsInterface` -> `ContrastsTable`

Public fields

`contrast_result` contrast results

`model_name` model name

`subject_id` default protein_Id

Methods**Public methods:**

- `ContrastsTable$new()`
- `ContrastsTable$get_contrast_sides()`
- `ContrastsTable$get_linfct()`
- `ContrastsTable$get_contrasts()`
- `ContrastsTable$get_Plotter()`
- `ContrastsTable$clone()`

Method `new()`: initialize

Usage:

```
ContrastsTable$new(
  contrastsdf,
  subject_id = "protein_Id",
  model_name = "ContrastTable"
)
```

Arguments:

`contrastsdf` data.frame

`subject_id` default protein_Id

`model_name` default ContrastTable

Method `get_contrast_sides()`: return sides of contrast

Usage:

```
ContrastsTable$get_contrast_sides()
```

Returns: data.frame

Method `get_linfct()`: not implemented

Usage:

```
ContrastsTable$get_linfct()
```

Method `get_contrasts()`: get contrasts

Usage:

```
ContrastsTable$get_contrasts(all = FALSE)
```

Arguments:

`all` should all columns be returned (default FALSE)

`global` use a global linear function (determined by `get_linfct`)

Method `get_Plotter()`: get [ContrastsPlotter](#)

Usage:

```
ContrastsTable$get_Plotter(fc_threshold = 1, fdr_threshold = 0.1)
```

Arguments:

`fc_threshold` fold change threshold

`fdr_threshold` fdr threshold

Returns: [ContrastsPlotter](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ContrastsTable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[summary_ROPECA_median_p_scaled](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```

istar <- prolfqua::sim_lfq_data_protein_config(Nprot = 30)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$rename_response("transformedIntensity")
Contrasts <- c("aC" = "group_A - group_Ctrl",
              "bC" = "group_B - group_Ctrl")
fa <- ContrastsLMFacade$new(lfqdata, "~ group_", Contrasts)
ctr <- fa$get_contrasts()
fa$contrast$subject_id
xcx <- ContrastsTable$new(ctr, subject_id = fa$contrast$subject_id, model_name = "TableTest")
xcx$get_contrasts()
xcx$get_Plotter()$volcano()
stopifnot(is.null(xcx$get_contrast_sides()))
stopifnot(is.null(xcx$get_linfct()))
stopifnot(ncol(xcx$to_wide()) == 9)

```

contrasts_linfct	<i>compute group averages</i>
------------------	-------------------------------

Description

used in p2621, p2109

Usage

```

contrasts_linfct(
  models,
  linfct,
  subject_id = "protein_Id",
  contrastfun = prolfqua::compute_lmer_contrast
)

```

Examples

```

modelSummary_A <- sim_build_models_lm()
m <- get_complete_model_fit(modelSummary_A$model_df)

linfct <- linfct_from_model(m$linear_model[[1]], as_list = FALSE)
factor_contrasts <- linfct_matrix_contrasts(linfct, c(A_vs_B = "TreatmentA - TreatmentB"))

factor_levelContrasts <- contrasts_linfct( m,
  factor_contrasts,
  subject_id = "protein_Id",
  contrastfun = prolfqua::compute_contrast)

```

`data_checksummarizationrobust87`*Reference input for the robust summarisation check (87 percent)*

Description

log2 peptide areas of one protein, used by the tests of the robust (medpolish/rlm) protein summarisation. See 'inst/script/README.md'.

Usage`data_checksummarizationrobust87`**Format**

A data frame with 20 rows and the columns 'log2Area', 'peptide_Id', 'sampleName'.

See Also

Other data: [PACKAGE_DATA](#), [data_checksummerobust69](#), [data_ionstar](#), [prolfqua_data\(\)](#), [x5463yzwer453bbb](#)

`data_checksummerobust69`*Reference input for the robust summarisation check (69 percent)*

Description

log2 peptide areas of one protein, used by the tests of the robust (medpolish/rlm) protein summarisation. See 'inst/script/README.md'.

Usage`data_checksummerobust69`**Format**

A data frame with 20 rows and the columns 'log2Area', 'peptide_Id', 'sampleName'.

See Also

Other data: [PACKAGE_DATA](#), [data_checksummarizationrobust87](#), [data_ionstar](#), [prolfqua_data\(\)](#), [x5463yzwer453bbb](#)

data_ionstar	<i>IonStar peptide-level example data</i>
--------------	---

Description

Peptide intensities of 200 proteins from the IonStar human cell lysate with E. coli spike-in dilution series (dilutions a to e, 4 replicates each), processed with MaxQuant. An R6 object holding the data, its [AnalysisConfiguration](#), and the methods `filtered()`, `normalized()` and `subset_normalized()`. See ‘inst/script/README.md’ for its origin.

Usage

data_ionstar

Format

An R6 object of class ‘IonstarData’; ‘\$data’ is a data frame with 26520 rows.

Source

PRIDE PXD003881

See Also

Other data: [PACKAGE_DATA](#), [data_checksummarizationrobust87](#), [data_checksummarizerobust69](#), [prolfqua_data\(\)](#), [x5463yzwer453bbb](#)

df.residual.rfit_prolfqua	<i>Residual degrees of freedom for an augmented rfit fit</i>
---------------------------	--

Description

Residual degrees of freedom for an augmented rfit fit

Usage

```
## S3 method for class 'rfit_prolfqua'
df.residual(object, ...)
```

Arguments

object	an ‘rfit_prolfqua’ fit
...	ignored

Value

number of observations minus number of coefficients.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_residues\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
fit <- strategy_rfit("Sepal.Length ~ Species")$model_fun(iris)
df.residual(fit)
```

df.residual.rlm	<i>Residual degrees of freedom for rlm objects</i>
-----------------	--

Description

stats::df.residual returns NA for `rlm` objects. This S3 method computes weighted residual df instead.

Usage

```
## S3 method for class 'rlm'
df.residual(object, ...)
```

Arguments

object	an <code>rlm</code> object
...	ignored

Value

numeric scalar

encode_bin_resp	<i>convert to binary response</i>
-----------------	-----------------------------------

Description

convert to binary response

Usage

```
encode_bin_resp(lfqdata, name = "bin_resp")
```

Arguments

lfqdata	LFQData object
name	column name for binary response

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
lfq$set_data(encode_bin_resp(lfq))
lfq$set_config_value("bin_resp", "bin_resp")
lfq$data_long()[["bin_resp"]]
```

estimate_intensity	<i>Aggregates e.g. protein abundances from peptide abundances</i>
--------------------	---

Description

Aggregates e.g. protein abundances from peptide abundances

Usage

```
estimate_intensity(lfqdata, method = c("medpolish", "rlm"))
```

Arguments

lfqdata	LFQData object
method	"medpolish" (Tukey's median polish, column 'medpolish') or "rlm" (robust regression with 'MASS::rlm', column 'lmrob')

Value

returns list with data (data.frame) and config (AnalysisConfiguration)

See Also

Other aggregation: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [aggregate_intensity_top_n\(\)](#), [plot_estimate\(\)](#), [plot_hierarchies_line\(\)](#)

Examples

```
dd <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(dd$data, dd$config)
lfq <- lfq$get_Transformer()$log2()$lfq
bbMed <- estimate_intensity(lfq, method = "medpolish")
bbRob <- estimate_intensity(lfq, method = "rlm")
nrow(bbMed$data)
nrow(bbRob$data)
xt <- dplyr::inner_join(bbMed$data, bbRob$data)
plot(xt$medpolish, xt$lmrob, log = "xy", pch = "*")
abline(0, 1, col = 2)
```

FACADE_REGISTRY

*Registry of available contrast facade classes***Description**

Read-only snapshot of the built-in prolfqua facade registry, taken at package load time. It is derived from the single source of truth (`.seed_facade_registry()`); use [register_facade](#) to add entries from downstream packages, [lookup_facade](#) to resolve a single entry, and [list_facades\(\)](#) to enumerate the live registry (which reflects downstream registrations).

Usage

FACADE_REGISTRY

Format

An object of class `facade_registry` (inherits from `list`) of length 18.

Details

Each entry has fields `class`, `needs`, `package`, and `needs_saint_annotation`.

Value

A named list of built-in facade registry entries.

Examples

```
names(FACADE_REGISTRY)
FACADE_REGISTRY$limma$class
```

```
filter_proteins_by_peptide_count
```

Keep only the proteins with enough peptides

Description

Keeps the rows of the proteins with at least 'config\$min_peptides_protein' children at the next hierarchy level, counted by [nr_B_in_A()], and adds that count. The package dataset 'data_ionstar' calls it from its 'filtered()' method.

Usage

```
filter_proteins_by_peptide_count(pdata, config)
```

Arguments

pdata	data.frame in the long format of 'config'
config	AnalysisConfiguration

Value

list with the filtered 'data' and the 'name' of the count column, 'NULL' when the hierarchy has no child level

See Also

Other preprocessing: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [apply_to_response_matrix\(\)](#), [get_rob scales\(\)](#), [nr_B_in_A\(\)](#), [robust_scale\(\)](#), [scale_with_subset\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()
filtered <- prolfqua::filter_proteins_by_peptide_count(istar$data, istar$config)
stopifnot(all(filtered$data[[filtered$name]] >= istar$config$min_peptides_protein))
```

```
find_d0_deqms
```

Find prior degrees of freedom for DEqMS

Description

Uses `limma::trigammaInverse` to estimate d0 from the mean residual variance after removing the count-dependent trend.

Usage

```
find_d0_deqms(mean_myfct)
```

Arguments

mean_myfct	mean of squared residuals minus trigamma correction
------------	---

Value

scalar prior degrees of freedom (d0)

find_package_file	<i>find file stored in package</i>
-------------------	------------------------------------

Description

find file stored in package

Usage

```
find_package_file(packagename, file)
```

Arguments

packagename	name of the R package
file	relative path to the file within the package

Value

The computed result.

Examples

```
find_package_file("prolfqua", "CITATION")
```

generate_contrasts	<i>Combined generate_contrasts</i>
--------------------	------------------------------------

Description

Combined generate_contrasts

Usage

```
generate_contrasts(primary_levels, secondary_levels, interactions = TRUE)
```

Arguments

primary_levels	character vector of primary factor levels
secondary_levels	character vector of secondary factor levels
interactions	logical, if TRUE include interaction contrasts

Value

Contrast definitions or contrast results.

See Also

Other contrasts: [annotation_add_contrasts\(\)](#), [generate_contrasts_for_factor\(\)](#), [interaction_contrasts\(\)](#), [level_specific_contrasts\(\)](#), [main_effect_contrasts\(\)](#)

Examples

```
primary_levels <- c("MI", "MINOCAM")
secondary_levels <- c("T0", "T150", "T300")
generate_contrasts(primary_levels, secondary_levels)
generate_contrasts(secondary_levels, primary_levels)
```

```
generate_contrasts_for_factor
```

Single-factor contrasts (pairwise comparisons)

Description

Single-factor contrasts (pairwise comparisons)

Usage

```
generate_contrasts_for_factor(levels)
```

Arguments

levels character vector of factor levels

Value

Contrast definitions or contrast results.

See Also

Other contrasts: [annotation_add_contrasts\(\)](#), [generate_contrasts\(\)](#), [interaction_contrasts\(\)](#), [level_specific_contrasts\(\)](#), [main_effect_contrasts\(\)](#)

Examples

```
# example code
primary_levels <- c("MI", "MINOCAM")
secondary_levels <- c("T0", "T150", "T300")
x2 <- level_specific_contrasts(primary_levels, secondary_levels)
x3 <- level_specific_contrasts(secondary_levels, primary_levels)
generate_contrasts_for_factor(names(x2))
```

get_anova_df	<i>anova returning dataframe</i>
--------------	----------------------------------

Description

anova returning dataframe

Usage

```
get_anova_df(test = "F")
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_2\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_residua\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```
x <- get_anova_df(test = "F")
x <- get_anova_df(test = "Chisq")
```

get_complete_model_fit	<i>retrieve complete models.</i>
------------------------	----------------------------------

Description

retrieve complete models.

Usage

```
get_complete_model_fit(complete_models)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
x <- sim_build_models_lmer(model = "factors", Nprot = 10)
cfits <- get_complete_model_fit(x$model_df)
stopifnot(nrow(cfits) == 6)
```

get_contrast	<i>Compute fold changes given Contrasts</i>
--------------	---

Description

Compute fold changes given Contrasts

Usage

```
get_contrast(data, hierarchy_keys, contrasts)
```

Arguments

- data hierarchy_keys of Analysis Configuration
- contrasts list of contrasts to compute

See Also

Other imputation: [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```

istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
lfq$complete_cases()

Contrasts <- c("dilution.b-a" = "group_A - group_B", "dilution.c-e" = "group_A - group_Ctrl")

var <- summarize_stats(lfq)
var <- prolfqua::make_interaction_column(var, columns = lfq$relevant_factor_keys())

imp <- var |> tidyr::pivot_wider(id_cols = lfq$hierarchy_keys(),
                              names_from = interaction,
                              values_from = !!rlang::sym("meanAbundance"))

imputed <- get_contrast(imp, lfq$hierarchy_keys(), Contrasts)

```

get_p_values_pbeta	<i>p-value of protein from p.value of the median fold change peptide.</i>
--------------------	---

Description

p-value of protein from p.value of the median fold change peptide.

Usage

```
get_p_values_pbeta(median.p.value, n.obs, max.n = 10)
```

Arguments

max.n limit number of peptides per protein.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpute](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#).

```
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()
```

Examples

```
plot(get_p_values_pbeta(0.1, 1:10, 10), ylim=c(0, 0.1))
plot(get_p_values_pbeta(0.1, 1:10, 3), ylim=c(0, 0.1))
plot(get_p_values_pbeta(0.3, 1:30, 3), ylim=c(0, 0.1))
abline(h=.05, col = 2)
plot(seq(0.0, 1.0, length=30), get_p_values_pbeta(seq(0.0, 1.0, length=30), rep(10, 30)))
abline(0, 1)
plot(seq(0.0, 1.0, length=30), get_p_values_pbeta(seq(0.0, 1.0, length=30), rep(10, 30), 3))
abline(0, 1)
testthat::expect_equal(get_p_values_pbeta(0.3, 10, 3), 0.216, tolerance = 1e-4)
testthat::expect_equal(get_p_values_pbeta(0, 10, 3), 0, tolerance = 1e-4)
testthat::expect_equal(get_p_values_pbeta(1, 10, 3), 1, tolerance = 1e-4)
testthat::expect_equal(get_p_values_pbeta(1, 10, 3), get_p_values_pbeta(1, 3, 10), tolerance = 1e-4)
```

get_rob scales

compute median and standard deviation for each sample

Description

compute median and standard deviation for each sample

Usage

```
get_rob scales(lfqdata)
```

See Also

Other preprocessing: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [apply_to_response_matrix\(\)](#), [filter_proteins_by_peptide_nr_B_in_A\(\)](#), [robust_scale\(\)](#), [scale_with_subset\(\)](#)

Examples

```
bb <- prolfqua::sim_lfq_data_peptide_config()
lfqdata <- prolfqua::LFQData$new(bb$data, bb$config)
lfqdata <- lfqdata$get_Transformer()$log2()$lfq
s1 <- get_rob scales(lfqdata)

res <- scale_with_subset(lfqdata, lfqdata)
lfqres <- prolfqua::LFQData$new(res$data, lfqdata$get_config())$clone(deep = TRUE)
s2 <- get_rob scales(lfqres)
abs(mean(s1$mads) - mean(s2$mads)) < 0.1
```

get_uniprot_id_from_fasta_header
Extracts uniprot ID

Description

Extracts uniprot ID

Usage

get_uniprot_id_from_fasta_header(df, idcolumn = "protein_Id")

Arguments

df data.frame
idcolumn character, column name to extract uniprot IDs from

Value

data.frame

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

Examples

```
df <- data.frame(  
  protein_Id = c("sp|P12345|PROT1_HUMAN", "sp|Q67890|PROT2_HUMAN",  
                 "tr|A0A0A0|PROT3_HUMAN", "NOFASTA_ID"),  
  value = 1:4  
)  
result <- get_uniprot_id_from_fasta_header(df, idcolumn = "protein_Id")  
stopifnot("UniprotID" %in% colnames(result))  
stopifnot(sum(!is.na(result$UniprotID)) == 3)
```

group_label *group label function*

Description

group label function

Usage

group_label(primary, secondary)

Arguments

primary	character, primary factor level
secondary	character, secondary factor level

Value

A character vector with combined group labels.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```
group_label("a", "b")
```

HierarchyCountsSample	<i>Count distinct elements for each level of hierarchy per sample</i>
-----------------------	---

Description

- Count distinct elements for each level of hierarchy per sample
- Count distinct elements for each level of hierarchy per sample

Details

Provides wide, long, and plot views of hierarchy counts.

Value

An R6 class generator.

Public fields

.summary summarised data frame
.configuration AnalysisConfiguration object

Methods**Public methods:**

- [HierarchyCountsSample\\$new\(\)](#)
- [HierarchyCountsSample\\$wide\(\)](#)
- [HierarchyCountsSample\\$long\(\)](#)
- [HierarchyCountsSample\\$plot\(\)](#)
- [HierarchyCountsSample\\$clone\(\)](#)

Method new(): Create a new HierarchyCountsSample

Usage:

```
HierarchyCountsSample$new(pdata, configuration, nr_children = 1)
```

Arguments:

pdata data frame
configuration AnalysisConfiguration
nr_children minimum number of children to include

Method wide(): Return wide-format summary

Usage:

```
HierarchyCountsSample$wide()
```

Method long(): Return long-format summary

Usage:

```
HierarchyCountsSample$long()
```

Method plot(): Return barplot of hierarchy counts

Usage:

```
HierarchyCountsSample$plot()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
HierarchyCountsSample$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other summary: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [hierarchy_counts\(\)](#), [summarize_hierarchy\(\)](#)

Examples

```
bb <- sim_lfq_data_protein_config()
counts <- HierarchyCountsSample$new(bb$data, bb$config)
head(counts$wide())
```

hierarchy_counts	<i>Count distinct elements for each level of hierarchy and isotope</i>
------------------	--

Description

E.g. number of proteins, peptides, precursors in the dataset

Usage

```
hierarchy_counts(pdata, hierarchy_keys, isotope_label = "isotopeLabel")
```

Arguments

```
pdata          data.frame
hierarchy_keys character vector — all hierarchy column names
isotope_label  character — isotope label column name
```

See Also

Other summary: [HierarchyCountsSample](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [summarize_hierarchy\(\)](#)

Examples

```
bb <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb$data, bb$config)

x <- hierarchy_counts(lfq$data_long(), lfq$hierarchy_keys(), lfq$isotope_label())
x$protein_Id
stopifnot(ncol(x) == length(lfq$hierarchy_keys()) + 1)
# select non existing protein
data0 <- lfq$data_long() |> dplyr::filter(protein_Id == "XYZ")
tmp <- hierarchy_counts(data0, lfq$hierarchy_keys(), lfq$isotope_label())
stopifnot(nrow(tmp) == 0)
```

impute_from_model	<i>Fill missing responses with predictions from per-subject linear models</i>
-------------------	---

Description

Completes the data to every subject x every sample and fills each missing response cell with `predict()` of that subject's fitted `lm`, on the response scale. Observed values are left unchanged. The imputed values carry no noise.

Usage

```
impute_from_model(model, lfqdata)
```

Arguments

```
model          a Model with lm fits, as returned by build\_model or build\_model\_impute
lfqdata        the LFQData the model was fitted on (same response and subject_id)
```

Details

Each subject gets one route, checked in this order:

complete no missing cell.

lod_refit the subject was refitted at the LOD by `build_model_impute` (`model_df$imputed`); that fit's predictions fill the missing cells.

fitted the observed-data fit has no NA coefficient and knows every factor level of the missing cells; its predictions fill them.

none no usable fit; the missing cells stay NA.

Value

list with `lfqdata`, a new `LFQData` holding every subject x sample row with missing responses filled, and `summary`, a tibble with the `subject_id` columns and `n_observed`, `n_imputed` and `route` per subject

See Also

`build_model_impute`

Other modelling: `AnovaExtractor`, `Contrasts`, `ContrastsDEqMSFacade`, `ContrastsDEqMSVoomFacade`, `ContrastsFacadeBase`, `ContrastsFirth`, `ContrastsFirthFacade`, `ContrastsFirthNestedFacade`, `ContrastsLMFacade`, `ContrastsLMImputeFacade`, `ContrastsLimma`, `ContrastsLimmaFacade`, `ContrastsLimmaImputeFacade`, `ContrastsLimmaVoomFacade`, `ContrastsLimmaVoomImputeFacade`, `ContrastsLimpaFacade`, `ContrastsLimpaNestedFacade`, `ContrastsLmerNestedFacade`, `ContrastsModerated`, `ContrastsModeratedDEqMS`, `ContrastsPlotter`, `ContrastsRLMFacade`, `ContrastsROPECA`, `ContrastsROPECANestedFacade`, `ContrastsRfitFacade`, `ContrastsRfitImputeFacade`, `ContrastsTable`, `INTERNAL_FUNCTIONS_BY_FAMILY`, `LR_test()`, `Model`, `ModelFirth`, `ModelLimma`, `StrategyLM`, `StrategyLimma`, `StrategyLimpa`, `StrategyLmer`, `StrategyLogistf`, `StrategyRLM`, `StrategyRfit`, `build_contrast_analysis()`, `build_model()`, `build_model_glm_peptide()`, `build_model_glm_protein()`, `build_model_impute()`, `build_model_limma()`, `build_model_limma_impute()`, `build_model_limma_voom()`, `build_model_limma_voom_impute()`, `build_model_limpa()`, `build_model_logistf()`, `compute_borrowed_variance()`, `compute_borrowed_variance_contrast()`, `compute_lmer_contrast()`, `df.residual.rfit_prolfqua()`, `get_anova_df()`, `get_complete_model_fit()`, `get_p_values_pbeta()`, `group_label()`, `impute_refit_singular()`, `linfct_from_model()`, `linfct_matrix_contrasts()`, `list_facades()`, `lookup_facade()`, `merge_contrasts_res`, `model_analyse()`, `model_summary()`, `moderated_p_deqms()`, `moderated_p_deqms_long()`, `moderated_p_limma()`, `new_imputed_model()`, `pivot_model_contrasts_to_wide()`, `register_facade()`, `sigma.rfit_prolfqua()`, `sim_build_models_lm()`, `sim_build_models_lmer()`, `sim_build_models_logistf()`, `sim_make_model_lm()`, `sim_make_model_lmer()`, `strategy_limma()`, `strategy_limpa()`, `strategy_logistf()`, `summary_ROPECA_median`, `vcov.rfit_prolfqua()`

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 30, weight_missing = 0.5)
lfqdata <- LFQData$new(istar$data, istar$config)
strat <- strategy_lm(paste(lfqdata$response(), "~ group-"))
mod <- build_model_impute(lfqdata, strat)
res <- impute_from_model(mod, lfqdata)
table(res$summary$route)
n_missing <- function(x) sum(is.na(x$data_long())$abundance))
stopifnot(n_missing(res$lfqdata) <= n_missing(lfqdata))
```

impute_refit_singular *Impute and refit singular/failed models*

Description

For proteins where the initial lm fit failed or produced NA coefficients, impute missing values with LOD, clamp, refit, and attach borrowed covariance.

Usage

```
impute_refit_singular(
  model_df,
  model_strategy,
  lod,
  response,
  sample_template,
  borrow_method = c("sigma", "vcov"),
  df_method = c("observed", "borrowed"),
  nr_children_col = NULL,
  on_misalign = c("fallback", "fail")
)
```

Arguments

model_df	tibble from model_analyse
model_strategy	strategy list from strategy_lm etc.
lod	numeric, limit of detection value
response	character, response column name in nested data
sample_template	data.frame with all sample/group combinations (columns matching the nested data minus the response). Used to complete proteins that are entirely missing in one or more groups.
borrow_method	"sigma" or "vcov"
df_method	"observed" uses max(n_observed - p, 1), "borrowed" uses median df from successful fits
nr_children_col	optional column name for nr_children (peptide counts); NA values in this column are filled with 1 for imputed rows so that weighted lm fits do not fail
on_misalign	passed to compute_borrowed_variance . With "fail", an unborrowable full covariance matrix skips the rescue (the model_df is returned unchanged) instead of crashing, so the affected proteins surface in get_missing().

Value

modified model_df with imputed models replacing failed/singular ones

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

`interaction_contrasts` *Interaction contrasts (difference of differences)*

Description

Interaction contrasts (difference of differences)

Usage

```
interaction_contrasts(primary_levels, secondary_levels)
```

Arguments

`primary_levels` character vector of primary factor levels
`secondary_levels` character vector of secondary factor levels

Value

Contrast definitions or contrast results.

See Also

Other contrasts: [annotation_add_contrasts\(\)](#), [generate_contrasts\(\)](#), [generate_contrasts_for_factor\(\)](#), [level_specific_contrasts\(\)](#), [main_effect_contrasts\(\)](#)

Examples

```
primary_levels <- c("MI", "MINOCAM")
secondary_levels <- c("T0", "T150", "T300")
interaction_contrasts(primary_levels, secondary_levels)
interaction_contrasts(secondary_levels, primary_levels)
```

INTERNAL_FUNCTIONS_BY_FAMILY

Internal Functions by category

Description

Internal Functions by category

Value

A documentation topic.

See Also

Other aggregation: [aggregate_intensity_top_n\(\)](#), [estimate_intensity\(\)](#), [plot_estimate\(\)](#), [plot_hierarchies_line\(\)](#)

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Other plotting: [ContrastsPlotter](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#)

Other preprocessing: [apply_to_response_matrix\(\)](#), [filter_proteins_by_peptide_count\(\)](#), [get_robscales\(\)](#), [nr_B_in_A\(\)](#), [robust_scale\(\)](#), [scale_with_subset\(\)](#)

Other stats: [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Other summary: [HierarchyCountsSample](#), [hierarchy_counts\(\)](#), [summarize_hierarchy\(\)](#)

Other utilities: [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

is_contaminant	<i>Detect contaminant identifiers</i>
----------------	---------------------------------------

Description

As [is_decoy](#), but for contaminant entries (keratin, trypsin, BSA, ...). Built-in anchored default prefixes unioned with an optional configured pattern; empty / NULL uses the defaults only.

Usage

```
is_contaminant(ids, pattern = NULL)
```

Arguments

ids	character vector of (prefixed) identifiers
pattern	optional additional contaminant regex, unioned with defaults

Value

logical vector, TRUE where an id looks like a contaminant

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

Examples

```
is_contaminant(c("zz|Cont00001|X", "sp|P2|X", "CON__ALBU"))
```

is_decoy	<i>Detect decoy / reverse-database identifiers</i>
----------	--

Description

Flags identifiers that look like decoy (reversed) database entries. Built-in anchored default prefixes are always considered, unioned with an optional configured pattern. An empty string or NULL pattern uses the defaults only.

Usage

```
is_decoy(ids, pattern = NULL)
```

Arguments

ids	character vector of (prefixed) identifiers
pattern	optional additional decoy regex, unioned with the defaults

Value

logical vector, TRUE where an id looks like a decoy

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

Examples

```
is_decoy(c("REV_sp|P1|X", "sp|P2|X", "decoy_3", "normalProtein"))
is_decoy(c("shuffled_1", "P2"), pattern = "^shuffled_")
```

level_specific_contrasts	<i>Level-specific contrasts (per secondary level)</i>
--------------------------	---

Description

Level-specific contrasts (per secondary level)

Usage

```
level_specific_contrasts(primary_levels, secondary_levels)
```

Arguments

primary_levels	character vector of primary factor levels
secondary_levels	character vector of secondary factor levels

Value

Contrast definitions or contrast results.

See Also

Other contrasts: [annotation_add_contrasts\(\)](#), [generate_contrasts\(\)](#), [generate_contrasts_for_factor\(\)](#), [interaction_contrasts\(\)](#), [main_effect_contrasts\(\)](#)

Examples

```
# example code

primary_levels <- c("MI", "MINOCAM")
secondary_levels <- c("T0", "T150", "T300")
x2 <- level_specific_contrasts(primary_levels, secondary_levels)
x3 <- level_specific_contrasts(secondary_levels, primary_levels)
```

LFQData	<i>LFQData R6 class</i>
---------	-------------------------

Description

LFQData R6 class

LFQData R6 class

Value

An R6 class generator.

Missing Data Assumptions

The filtering and imputation methods in this package assume that missing values are Missing Completely At Random (MCAR) or Missing At Random (MAR). Abundance-dependent missingness (MNAR), which is common in DDA proteomics, is not modelled. Users should be aware that MNAR can bias fold-change estimates and inflate false discovery rates.

Public fields

prefix e.g. "peptide_", "protein_", "compound_"

Methods**Public methods:**

- [LFQData\\$new\(\)](#)
- [LFQData\\$set_data\(\)](#)
- [LFQData\\$get_config\(\)](#)
- [LFQData\\$set_config_value\(\)](#)
- [LFQData\\$get_copy\(\)](#)
- [LFQData\\$get_sample\(\)](#)
- [LFQData\\$get_subset\(\)](#)

- LFQData\$subject_id()
- LFQData\$is_transformed()
- LFQData\$remove_small_intensities()
- LFQData\$filter_proteins_by_peptide_count()
- LFQData\$remove_decoys()
- LFQData\$decoy_proportion()
- LFQData\$complete_cases()
- LFQData\$data_wide()
- LFQData\$factors()
- LFQData\$hierarchy()
- LFQData\$response()
- LFQData\$hierarchy_keys()
- LFQData\$relevant_hierarchy_keys()
- LFQData\$factor_keys()
- LFQData\$relevant_factor_keys()
- LFQData\$sample_name()
- LFQData\$file_name()
- LFQData\$nr_children_col()
- LFQData\$isotope_label()
- LFQData\$data_long()
- LFQData\$rename_response()
- LFQData\$hierarchy_counts()
- LFQData\$summarize_hierarchy()
- LFQData\$get_Plotter()
- LFQData\$get_Summariser()
- LFQData\$get_Stats()
- LFQData\$get_Transformer()
- LFQData\$get_Aggregator()
- LFQData\$clone()

Method new(): initialize

Usage:

```
LFQData$new(data, config, prefix = "ms_", setup = FALSE)
```

Arguments:

data data.frame

config configuration

prefix will be use as output prefix

setup is data setup needed, default = FALSE, if TRUE, calls [setup_analysis](#) on data first.

Method set_data(): set data (replaces the internal data frame)

Usage:

```
LFQData$set_data(new_data)
```

Arguments:

new_data data.frame

Returns: self (invisible)

Method `get_config()`: return the AnalysisConfiguration object

Usage:

`LFQData$get_config()`

Returns: AnalysisConfiguration

Method `set_config_value()`: set a config field value

Usage:

`LFQData$set_config_value(field, value)`

Arguments:

field character — field name

value the value to set

Method `get_copy()`: get deep copy

Usage:

`LFQData$get_copy()`

Method `get_sample()`: samples subset of data

Usage:

`LFQData$get_sample(size = 100, seed = NULL)`

Arguments:

size size of subset default 100

seed set seed

Method `get_subset()`: get subset of data

Usage:

`LFQData$get_subset(x)`

Arguments:

x data frame with columns containing subject_id

Method `subject_id()`: get subject ID columns

Usage:

`LFQData$subject_id()`

Method `is_transformed()`: is data transformed

Usage:

`LFQData$is_transformed(is_transformed)`

Arguments:

is_transformed logical

Returns: logical

Method `remove_small_intensities()`: some software is reporting NA's as 0, you must remove it from your data

Usage:

`LFQData$remove_small_intensities(threshold = 4)`

Arguments:

threshold default 4.

Returns: self

Method filter_proteins_by_peptide_count(): remove proteins with less than X peptides

Usage:

```
LFQData$filter_proteins_by_peptide_count()
```

Returns: self

Method remove_decoys(): remove decoy / reverse-database entries (rows whose top-level hierarchy id, e.g. protein_Id, is a decoy). Detection uses the configured 'pattern_decoys' unioned with the built-in defaults (see [is_decoy](#)); decoys are always detectable even when no pattern is configured. Returns a new decoy-free LFQData (self is not modified).

Usage:

```
LFQData$remove_decoys()
```

Returns: LFQData without decoys

Method decoy_proportion(): proportion of modelling-level keys (subject_id, e.g. protein or peptide by 'hierarchy_depth') that are decoys — an empirical-FDR / target-decoy QC signal. Decoy status derives from the top-level hierarchy id (protein_Id).

Usage:

```
LFQData$decoy_proportion()
```

Returns: numeric in [0, 1]

Method complete_cases(): some software is reporting NA's as 0, you must remove it from your data

Usage:

```
LFQData$complete_cases()
```

Arguments:

threshold default 4.

Returns: self

Method data_wide(): converts the data to wide

Usage:

```
LFQData$data_wide(as.matrix = FALSE, value = NULL)
```

Arguments:

as.matrix return as data.frame or matrix

value see possible lfqdata\$get_config()\$value_vars()

Returns: list with data, annotation, and configuration

Method factors(): Annotation table

Usage:

```
LFQData$factors()
```

Returns: data.frame

Method hierarchy(): Hierarchy table

Usage:

```
LFQData$hierarchy()
```

Method response(): name of response variable

Usage:

```
LFQData$response()
```

Returns: data.frame

Method hierarchy_keys(): return all hierarchy column names

Usage:

```
LFQData$hierarchy_keys()
```

Method relevant_hierarchy_keys(): return hierarchy column names at current depth (alias for subject_id)

Usage:

```
LFQData$relevant_hierarchy_keys()
```

Method factor_keys(): return all factor column names

Usage:

```
LFQData$factor_keys()
```

Method relevant_factor_keys(): return factor column names at current depth

Usage:

```
LFQData$relevant_factor_keys()
```

Method sample_name(): return sample name column

Usage:

```
LFQData$sample_name()
```

Method file_name(): return file name column

Usage:

```
LFQData$file_name()
```

Method nr_children_col(): return name of nr_children column

Usage:

```
LFQData$nr_children_col()
```

Method isotope_label(): return isotope label column name

Usage:

```
LFQData$isotope_label()
```

Method data_long(): return the tidy (long-format) data frame

Usage:

```
LFQData$data_long(na.omit = FALSE)
```

Arguments:

na.omit if TRUE, remove rows with NA in response column

Method rename_response(): new name of response variable

Usage:

```
LFQData$rename_response(newname = "Intensity")
```

Arguments:

newname default Intensity

Method `hierarchy_counts()`: number of elements at each level

Usage:

`LFQData$hierarchy_counts()`

Method `summarize_hierarchy()`: e.g. number of peptides per protein etc

Usage:

`LFQData$summarize_hierarchy()`

Returns: data.frame

Method `get_Plotter()`: get Plotter

Usage:

`LFQData$get_Plotter()`

Returns: LFQDataPlotter

Method `get_Summariser()`: get Summariser

Usage:

`LFQData$get_Summariser()`

Returns: LFQDataSummarizer

Method `get_Stats()`: Get [LFQDataStats](#). For more details see [LFQDataStats](#).

Usage:

`LFQData$get_Stats(stats = c("everything", "interaction", "all"))`

Arguments:

stats default interaction, computes statistics within interaction.

Returns: LFQDataStats

Method `get_Transformer()`: get Stats

Usage:

`LFQData$get_Transformer()`

Returns: LFQDataTransformer

Method `get_Aggregator()`: get Aggregator

Usage:

`LFQData$get_Aggregator(method = "medpolish", ...)`

Arguments:

method aggregation method: "medpolish", "rlm", or "topN"

... passed to aggregator constructor (e.g. prefix, N, func)

Returns: AggregateMedpolish, AggregateRlm, or AggregateTopN

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`LFQData$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```

istar <- sim_lfq_data_peptide_config()
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata$filter_proteins_by_peptide_count()
tmp <- lfqdata$data_wide()
testthat::expect_equal(nrow(tmp$data) , nrow(tmp$rowdata))
testthat::expect_equal(ncol(tmp$data) , nrow(tmp$annotation) + ncol(tmp$rowdata))

stopifnot("data.frame" %in% class(tmp$data))
tmp <- lfqdata$data_wide(as.matrix = TRUE)
stopifnot("matrix" %in% class(tmp$data))
stopifnot(lfqdata$is_transformed()==FALSE)
lfqdata$summarize_hierarchy()

lfqdata$response()
lfqdata$rename_response("peptide.intensity")
lfqdata$response()
stopifnot("LFQData" %in% class(lfqdata$get_copy()))
stopifnot("LFQDataTransformer" %in% class(lfqdata$get_Transformer()))
stopifnot("LFQDataStats" %in% class(lfqdata$get_Stats()))
stopifnot("LFQDataSummariser" %in% class(lfqdata$get_Summariser()))
stopifnot("LFQDataPlotter" %in% class(lfqdata$get_Plotter()))
stopifnot("AggregateMedpolish" %in% class(lfqdata$get_Aggregator("medpolish")))

tmp <-lfqdata$get_sample(5, seed = 4)
stopifnot(nrow(tmp$hierarchy()) == 5)

```

LFQDataPlotter

LFQDataPlotter — Create various visualization of the LFQdata

Description

LFQDataPlotter — Create various visualization of the LFQdata

LFQDataPlotter — Create various visualization of the LFQdata

Value

An R6 class generator.

Public fields

lfq LFQData object

prefix prefix to figure names when writing, e.g. protein_

Methods

Public methods:

- `LFQDataPlotter$new()`
- `LFQDataPlotter$raster()`
- `LFQDataPlotter$heatmap()`
- `LFQDataPlotter$heatmap_cor()`
- `LFQDataPlotter$pca()`
- `LFQDataPlotter$pca_plotly()`
- `LFQDataPlotter$boxplots()`
- `LFQDataPlotter$missigness_histogram()`
- `LFQDataPlotter$na_heatmap()`
- `LFQDataPlotter$intensity_distribution_density()`
- `LFQDataPlotter$intensity_distribution_violin()`
- `LFQDataPlotter$pairs_smooth()`
- `LFQDataPlotter$sample_correlation()`
- `LFQDataPlotter$upset_missing()`
- `LFQDataPlotter$write_boxplots()`
- `LFQDataPlotter$write_pdf()`
- `LFQDataPlotter$clone()`

Method `new()`: create LFQDataPlotter

Usage:

```
LFQDataPlotter$new(lfqdata, prefix = "ms_")
```

Arguments:

`lfqdata` LFQData

`prefix` will be prepended to outputs written

Method `raster()`: plot intensities in raster

Usage:

```
LFQDataPlotter$raster(
  arrange = c("mean", "var"),
  not_na = FALSE,
  rownames = FALSE,
  max_rownames_chars = 60,
  max_sample_label_chars = 20
)
```

Arguments:

`arrange` arrange by either mean or var

`not_na` TRUE arrange by number of NA's, FALSE by arrange by intensity

`rownames` show rownames (default FALSE - do not show.)

`max_rownames_chars` maximum displayed row label length

`max_sample_label_chars` maximum displayed sample label length. Labels keep their suffix because sample prefixes are often shared.

Returns: ComplexHeatmap::Heatmap

Method `heatmap()`: heatmap of intensities - columns are samples, rows are proteins or peptides. The abundances of each protein (row) are z-scored. Afterward, the mean abundance for each protein is zero, and the standard variation is one. z-scoring allows to compare (cluster) the proteins according to the difference in the expression in the samples. Without the z-scoring, the proteins would group according to their abundance, e.g., high abundant proteins would be one cluster.

Only the `top_n` most variable features are shown. Row clustering uses `stats::hclust`, which errors above 65536 features, so the rows are ranked by their variability statistic (CV for untransformed data, sd for transformed data; see [LFQDataStats](#)) and the most variable are kept. This keeps the heatmap feasible and legible for peptide-list / entrapment searches with tens of thousands of features.

Usage:

```
LFQDataPlotter$heatmap(
  na_fraction = 1,
  rownames = FALSE,
  max_rownames_chars = 60,
  max_sample_label_chars = 20,
  top_n = 1000
)
```

Arguments:

`na_fraction` maximum fraction of missing values per row. The default keeps every row that can be meaningfully z-scored.

`rownames` show rownames (default FALSE - do not show.)

`max_rownames_chars` maximum displayed row label length

`max_sample_label_chars` maximum displayed sample label length. Labels keep their suffix because sample prefixes are often shared.

`top_n` keep the `top_n` most variable features (default 1000); NULL or Inf keeps all features.

Returns: ComplexHeatmap::Heatmap

Method `heatmap_cor()`: heatmap of sample correlations.

The Spearman correlation among all samples is computed. Then the euclidean distance is used to compute the distances.

Usage:

```
LFQDataPlotter$heatmap_cor(max_sample_label_chars = 20)
```

Arguments:

`max_sample_label_chars` maximum displayed sample label length. Labels keep their suffix because sample prefixes are often shared.

Returns: ComplexHeatmap::Heatmap

Method `pca()`: PCA plot

A PCA is applied and the first and second principal component are shown. Features with missing values are removed. To keep all features, impute first, e.g. with [AggregatElimpa](#) in `impute_only` mode.

Usage:

```
LFQDataPlotter$pca(PC = c(1, 2), add_txt = TRUE, nudge = 0.1)
```

Arguments:

`PC` default `c(1,2)` - first and second principal component

`add_txt` show sample names

nudge default 0.1 nudge point labels

Returns: ggplot

Method `pca_plotly()`: pca plot

Usage:

```
LFQDataPlotter$pca_plotly(PC = c(1, 2), add_txt = FALSE)
```

Arguments:

PC default c(1,2) - first and second principal component

add_txt show sample names

Returns: plotly

Method `boxplots()`: boxplots for all proteins

Usage:

```
LFQDataPlotter$boxplots(facet = TRUE)
```

Arguments:

facet enable facet wrap if hierarchy_depth less then hierarchy length.

Returns: tibble with column boxplots containing ggplot objects

Method `missigness_histogram()`: histogram of intensities given number of missing in conditions

Usage:

```
LFQDataPlotter$missigness_histogram()
```

Returns: ggplot

Method `na_heatmap()`: heatmap of features with missing values

Usage:

```
LFQDataPlotter$na_heatmap()
```

Returns: ComplexHeatmap::Heatmap

Method `intensity_distribution_density()`: density distribution of intensities

Usage:

```
LFQDataPlotter$intensity_distribution_density(
  legend = NA,
  max_legend_samples = 16
)
```

Arguments:

legend show legend TRUE, FALSE do not show, NA selects automatically based on sample count.

max_legend_samples maximum number of samples for automatic legend display.

Returns: ggplot

Method `intensity_distribution_violin()`: Violinplot showing distribution of intensities in all samples

Usage:

```
LFQDataPlotter$intensity_distribution_violin()
```

Returns: ggplot

Method pairs_smooth(): pairsplot of intensities

Usage:

```
LFQDataPlotter$pairs_smooth(max = 10)
```

Arguments:

max maximal number of samples to show

Returns: NULL

Method sample_correlation(): plot of sample correlations

Usage:

```
LFQDataPlotter$sample_correlation()
```

Returns: NULL

Method upset_missing(): upset plot based on presence absence information

Usage:

```
LFQDataPlotter$upset_missing()
```

Returns: plot

Method write_boxplots(): write boxplots to file

Usage:

```
LFQDataPlotter$write_boxplots(path_qc, filename = NULL, width = 6, height = 6)
```

Arguments:

path_qc path to write to

filename file to write into

width fig width

height fig height

Method write_pdf(): write figure to pdf

Usage:

```
LFQDataPlotter$write_pdf(fig, path_qc, fig_name, width = 7, height = 7)
```

Arguments:

fig ggplot or ComplexHeatmap::Heatmap

path_qc path to write to

fig_name name of figure (no extension)

width figure width

height figure height

Returns: path the file was written to

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
LFQDataPlotter$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also[plot_heatmap_cor](#)[plot_pca](#)

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataStats](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()

lfqdata <- LFQData$new(
  istar$data,
  istar$config)
lfqplotter <- lfqdata$get_Plotter()

stopifnot(methods::is(lfqplotter$heatmap(), "Heatmap"))
stopifnot(methods::is(lfqplotter$heatmap_cor(), "Heatmap"))
stopifnot("ggplot" %in% class(lfqplotter$pca()))
stopifnot("plotly" %in% class(lfqplotter$pca_plotly()))
tmp <- lfqplotter$boxplots()
stopifnot("ggplot" %in% class(tmp$boxplot[[1]]))
stopifnot("ggplot" %in% class(lfqplotter$missigness_histogram()))

stopifnot(methods::is(lfqplotter$na_heatmap(), "Heatmap"))
class(lfqplotter$intensity_distribution_density())
class(lfqplotter$intensity_distribution_violin())
stopifnot(is.null(lfqplotter$pairs_smooth()))
stopifnot(class(lfqplotter$sample_correlation()) == "list")
stopifnot(methods::is(lfqplotter$raster(), "Heatmap"))
stopifnot("upset" == class(lfqplotter$upset_missing()))
wide <- lfqdata$data_wide(as.matrix = TRUE)
stopifnot(class(prolfqua::plot_sample_correlation(wide$data)) == "list")
```

LFQDataStats

*Decorates LFQData with methods to compute statistics of interactions***Description**

Decorates LFQData with methods to compute statistics of interactions

Decorates LFQData with methods to compute statistics of interactions

Details

compute stdv, mean and CV per peptide or protein and condition.

Value

An R6 class generator.

Public fields

lfq LFQData
 stat either CV or sd (if is_transformed)
 statsdf frame with statistics.

Methods**Public methods:**

- LFQDataStats\$new()
- LFQDataStats\$stats()
- LFQDataStats\$stats_wide()
- LFQDataStats\$stats_quantiles()
- LFQDataStats\$density()
- LFQDataStats\$density_median()
- LFQDataStats\$violin()
- LFQDataStats\$violin_median()
- LFQDataStats\$stdv_vs_mean()
- LFQDataStats\$power_t_test_quantiles()
- LFQDataStats\$power_t_test()
- LFQDataStats\$clone()

Method new(): create analyse variances and CV

Usage:

```
LFQDataStats$new(lfqdata, stats = c("everything", "interaction", "all"))
```

Arguments:

lfqdata LFQData object
 stats if 'interaction', compute within-group stats; if 'all', compute overall CV; if 'pooled', use pooled variance with grouping information

Method stats(): access data.frame with statistics

Usage:

```
LFQDataStats$stats()
```

Returns: data.frame with computed statistics

Method stats_wide(): access data.frame with statistics in wide format

Usage:

```
LFQDataStats$stats_wide()
```

Returns: data.frame with computed statistics in wide format

Method stats_quantiles(): Determine CV or sd for the quantiles

Usage:

```
LFQDataStats$stats_quantiles(probs = c(0.1, 0.25, 0.5, 0.75, 0.9))
```

Arguments:

probs for which quantile to determine CV or sd

Method density(): plots density or ecdf

Usage:

```
LFQDataStats$density(ggstat = c("density", "ecdf"))
```

Arguments:

ggstat either density or ecdf

Returns: ggplot

Method density_median(): plot density or ecdf of CV or sd for the 50

Usage:

```
LFQDataStats$density_median(ggstat = c("density", "ecdf"))
```

Arguments:

ggstat either density or ecdf

Returns: ggplot

Method violin(): plot violinplot of CV or sd

Usage:

```
LFQDataStats$violin()
```

Arguments:

ggstat either density or ecdf

Returns: ggplot

Method violin_median(): plot violinplot of CV or sd for the 50

Usage:

```
LFQDataStats$violin_median()
```

Returns: ggplot

Method stdv_vs_mean(): plot sd vs mean

Usage:

```
LFQDataStats$stdv_vs_mean(size = 200)
```

Arguments:

size number of points to sample (default 200)

Returns: ggplot

Method power_t_test_quantiles(): compute sample size for entire dataset

Usage:

```
LFQDataStats$power_t_test_quantiles(
  probs = c(0.1, 0.25, 0.5, 0.75, 0.9),
  delta = c(0.59, 1, 2),
  power = 0.8,
  sig.level = 0.05
)
```

Arguments:

probs quantiles of sd for which sample size should be computed

delta effect size

power power of test

sig.level significance level.

Method `power_t_test()`: compute sample for each protein

Usage:

```
LFQDataStats$power_t_test(delta = c(0.59, 1, 2), power = 0.8, sig.level = 0.05)
```

Arguments:

`delta` effect size

`power` power of test

`sig.level` significance level.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
LFQDataStats$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataSummariser](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```
# study variance of not normalized data
#source("c:/Users/wewol/prog/prolfqua/R/LFQData.R")
runallfuncs <- function(x){

  stopifnot("data.frame" %in% class(x$stats()))
  stopifnot("data.frame" %in% class(x$stats_wide()))
  stopifnot(c("long", "wide") %in% names(x$stats_quantiles()))
  stopifnot("ggplot" %in% class(x$density()))
  stopifnot("ggplot" %in% class(x$density_median()))
  stopifnot("ggplot" %in% class(x$density("ecdf")))
  stopifnot("ggplot" %in% class(x$density_median("ecdf")))
  stopifnot("ggplot" %in% class(x$violin()))
  stopifnot("ggplot" %in% class(x$violin_median()))
  stopifnot("ggplot" %in% class(x$stdv_vs_mean(size = 400)))
  if(!x$lfq$is_transformed()){
    stopifnot(is.null(x$power_t_test()))
    stopifnot(is.null(x$power_t_test_quantiles()))
  }
}
bb <- prolfqua::sim_lfq_data_peptide_config()

lfqdata <- LFQData$new(bb$data, bb$config)
lfqstats <- lfqdata$get_Stats()
stopifnot(ncol(lfqstats$stats_wide()) == 30)
lfqstats$violin()
runallfuncs(lfqstats)
x <- lfqstats

#study variance of normalized data
```

```

lfqdata <- LFQData$new(bb$data, bb$config)
lfqdata$transformed(TRUE)
lfqstats <- lfqdata$get_Stats()
stopifnot(ncol(lfqstats$stats_wide()) == 26)
runallfuncs(lfqstats)

#Slightly different dataset

# estimates statistics for all samples
lfqstats <- lfqdata$get_Stats(stats = "all")
stopifnot(ncol(lfqstats$stats_wide()) == 8)
runallfuncs(lfqstats)
lfqstats <- lfqdata$get_Stats(stats = "interaction")
stopifnot(ncol(lfqstats$stats_wide()) == 20)
runallfuncs(lfqstats)

# Group size 1
bb <- prolfqua::sim_lfq_data_peptide_config(N=1)
lfq1 <- LFQData$new(bb$data, bb$config)
table_factors_size(lfq1$data_long(), lfq1$file_name(),
  lfq1$sample_name(), lfq1$factor_keys(), lfq1$relevant_factor_keys())
lfqdata <- LFQData$new(bb$data, bb$config)
lfqstats <- lfqdata$get_Stats()

# stopifnot(ncol(lfqstats$stats_wide()) == 30)
lfqstats$violin()
runallfuncs(lfqstats)

```

LFQDataSummariser

Summarize LFQData

Description

Summarize LFQData

Summarize LFQData

Value

An R6 class generator.

Public fields

lfq LFQData

Methods

Public methods:

- [LFQDataSummariser\\$new\(\)](#)
- [LFQDataSummariser\\$hierarchy_counts\(\)](#)
- [LFQDataSummariser\\$hierarchy_counts_sample\(\)](#)
- [LFQDataSummariser\\$plot_hierarchy_counts_sample\(\)](#)

- `LFQDataSummariser$interaction_missing_stats()`
- `LFQDataSummariser$upset_interaction_missing_stats()`
- `LFQDataSummariser$missingness_per_group()`
- `LFQDataSummariser$missingness_per_group_cumsum()`
- `LFQDataSummariser$plot_missingness_per_group()`
- `LFQDataSummariser$plot_missingness_per_group_cumsum()`
- `LFQDataSummariser$percentage_abundance()`
- `LFQDataSummariser$clone()`

Method `new()`: initialize

Usage:

`LFQDataSummariser$new(lfqdata)`

Arguments:

`lfqdata` LFQData

Method `hierarchy_counts()`: summarize hierarchy

Usage:

`LFQDataSummariser$hierarchy_counts()`

Method `hierarchy_counts_sample()`: number of elements at each level in every sample

Usage:

```
LFQDataSummariser$hierarchy_counts_sample(
  value = c("wide", "long"),
  nr_children = 1
)
```

Arguments:

`value` wide - wide format, long - long format, plot - ggplot

`nr_children` get summary for 1,2 or more number of children

Method `plot_hierarchy_counts_sample()`: barplot showing number of elements at each level in every sample

Usage:

`LFQDataSummariser$plot_hierarchy_counts_sample(nr_children = 1)`

Arguments:

`nr_children` get summary for 1,2 or more number of children

`value` wide - wide format, long - long format, plot - ggplot

Method `interaction_missing_stats()`: missing per condition and protein

Usage:

`LFQDataSummariser$interaction_missing_stats()`

Method `upset_interaction_missing_stats()`: upset plot with missing information per protein and condition

Usage:

`LFQDataSummariser$upset_interaction_missing_stats(tr = 2)`

Arguments:

`tr` if less than tr observations in condition then missing

Method `missingness_per_group()`: missing stats per condition

Usage:

```
LFQDataSummariser$missingness_per_group()
```

Method `missingness_per_group_cumsum()`: missing stats per condition as cumulative sum

Usage:

```
LFQDataSummariser$missingness_per_group_cumsum()
```

Method `plot_missingness_per_group()`: barplot with number of features with 1,2, etc missing in condition

Usage:

```
LFQDataSummariser$plot_missingness_per_group()
```

Returns: ggplot

Method `plot_missingness_per_group_cumsum()`: barplot with cumulative sum of features with 1,2, etc missing in condition

Usage:

```
LFQDataSummariser$plot_missingness_per_group_cumsum()
```

Returns: ggplot

Method `percentage_abundance()`: Does roll up to highest hierarchy and Computes the percent abundance of proteins overall and within each group

Usage:

```
LFQDataSummariser$percentage_abundance()
```

Returns: data frame

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
LFQDataSummariser$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataToSummarizedExperiment\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()

data <- istar$data
lfqdata <- LFQData$new(data, istar$config)
sum <- lfqdata$get_Summariser()
sum
sum$hierarchy_counts()
sum$hierarchy_counts_sample("wide")
sum$hierarchy_counts_sample("long")
sum$plot_hierarchy_counts_sample()
sum$plot_hierarchy_counts_sample()
```

```
tmp <- sum$interaction_missing_stats()

sum$missingness_per_group()
sum$missingness_per_group_cumsum()
sum$plot_missingness_per_group()
sum$plot_missingness_per_group_cumsum()
sum$upset_interaction_missing_stats()
sum$percentage_abundance()
```

LFQDataToSummarizedExperiment

converts LFQData object to SummarizedExperiment

Description

For compatibility with Bioconductor

Usage

```
LFQDataToSummarizedExperiment(lfqdata)
```

Arguments

lfqdata LFQData object

Value

SummarizedExperiment (bioconductor)

See Also

Other LFQData: [AggregateLimpa](#), [AggregateMedpolish](#), [AggregateRlm](#), [AggregateTopN](#), [AggregatorBase](#), [LFQData](#), [LFQDataPlotter](#), [LFQDataStats](#), [LFQDataSummariser](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()
data <- istar$data
lfqdata <- LFQData$new(data, istar$config)
lfqdata$data_wide()
if(require("SummarizedExperiment")){
  tmp <- LFQDataToSummarizedExperiment(lfqdata)
}
```

LFQDataTransformer	<i>Decorate LFQData with Methods for transforming Intensities</i>
--------------------	---

Description

Decorate LFQData with Methods for transforming Intensities

Decorate LFQData with Methods for transforming Intensities

Value

An R6 class generator.

Public fields

lfq LFQData object

Methods**Public methods:**

- `LFQDataTransformer$new()`
- `LFQDataTransformer$log2()`
- `LFQDataTransformer$get_scales()`
- `LFQDataTransformer$robscale()`
- `LFQDataTransformer$robscale_subset()`
- `LFQDataTransformer$center_to_reference()`
- `LFQDataTransformer$intensity_array()`
- `LFQDataTransformer$intensity_matrix()`
- `LFQDataTransformer$clone()`

Method `new()`: initialize

Usage:

`LFQDataTransformer$new(lfqdata)`

Arguments:

lfqdata LFQData object to transform

Method `log2()`: log2 transform data

Usage:

`LFQDataTransformer$log2(force = FALSE)`

Arguments:

force if 'FALSE', data already log2 transformed will not be transformed a second time. 'TRUE' forces log transformation

Returns: LFQDataTransformer

Method `get_scales()`: get mean and variance and standard deviation in each sample

Usage:

`LFQDataTransformer$get_scales()`

Returns: list with means and mads

Method `robscale()`: robust scale data

Usage:

```
LFQDataTransformer$robscale(
  preserve_mean = TRUE,
  colname = "transformedIntensity"
)
```

Arguments:

`preserve_mean` should original mean value be preserved TRUE, if FALSE then center at zero
`colname` new name of transformed column

Returns: LFQDataTransformer (self)

Method `robscale_subset()`: log2 transform and robust scale data based on subset

Usage:

```
LFQDataTransformer$robscale_subset(
  lfqsubset,
  preserve_mean = TRUE,
  colname = "transformed_abundance"
)
```

Arguments:

`lfqsubset` LFQData subset to use for normalization
`preserve_mean` should original mean value be preserved TRUE, if FALSE then center at zero
`colname` - how to name the transformed intensities, default transformedIntensity

Returns: LFQDataTransformer (self)

Method `center_to_reference()`: center data to a reference subset

Usage:

```
LFQDataTransformer$center_to_reference(lfqsubset)
```

Arguments:

`lfqsubset` LFQData subset to use as reference

Returns: LFQDataTransformer (self)

Method `intensity_array()`: Transforms intensities

Usage:

```
LFQDataTransformer$intensity_array(.func = log2, force = FALSE)
```

Arguments:

`.func` transformation function working with arrays e.g. log2, log10, asinh etc.
`force` transformation on already transformed data.

Returns: LFQDataTransformer (self)

Method `intensity_matrix()`: pass a function which works with matrices, e.g., `vsn::justvsn`

Usage:

```
LFQDataTransformer$intensity_matrix(.func = robust_scale, force = FALSE)
```

Arguments:

.func any function taking a matrix and returning a matrix (columns sample, rows feature, e.g. 'base::scale()'), default 'robust_scale'
 force transformation on data already transformed
Returns: LFQDataTransformer (self)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
LFQDataTransformer$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
istar <- sim_lfq_data_peptide_config(Nprot = 50)
lfqdata <- LFQData$new(istar$data, istar$config)

lfqcopy <- lfqdata$get_copy()
lfqTrans <- lfqcopy$get_Transformer()

x <- lfqTrans$intensity_array(log2)

x$lfq$is_transformed()
x <- x$intensity_matrix(robust_scale)
plotter <- x$lfq$get_Plotter()
plotter$intensity_distribution_density()

# transform by asinh root and scale

lfqcopy <- lfqdata$get_copy()
lfqTrans <- lfqcopy$get_Transformer()
x <- lfqTrans$intensity_array(asinh)
mads1 <- mean(x$get_scales())$mads)
x <- lfqTrans$intensity_matrix(robust_scale, force = TRUE)
mads2 <- mean(x$get_scales())$mads)
stopifnot(abs(mads1 - mads2) < 1e-8)

stopifnot(abs(mean(x$get_scales())$medians)) < 1e-8)
lfqcopy <- lfqdata$get_copy()
lfqTrans <- lfqcopy$get_Transformer()
lfqTrans$log2()
before <- lfqTrans$get_scales()
lfqTrans$robscale()
after <- lfqTrans$get_scales()
stopifnot(abs(mean(before$medians) - mean(after$medians)) < 1e-8)
stopifnot(abs(mean(before$mads) - mean(after$mads)) < 1e-8)

# normalize data using vsn
lfqcopy <- lfqdata$get_copy()
lfqTrans <- lfqcopy$get_Transformer()
lfqTransCheck <- lfqcopy$get_Transformer()

lfqTransCheck$log2()
lfqTransCheck$get_scales()
lfqTransCheck$lfq$get_Plotter()$intensity_distribution_density()
```

```

if(require("vsn")){
  res <- lfqTrans$intensity_matrix( .func = vsn::justvsn)
  res$lfq$get_Plotter()$intensity_distribution_density()
  res$get_scales()
}
if(require("preprocessCore")){
  quant <- function(y){
    ynorm <- preprocessCore::normalize.quantiles(y)
    rownames(ynorm) <- rownames(y)
    colnames(ynorm) <- colnames(y)
    return(ynorm)
  }
  res <- lfqTrans$intensity_matrix( .func = quant)
  res$lfq$get_Plotter()$intensity_distribution_density()
}

#subset scaling

istar2 <- sim_lfq_data_peptide_config()
lfqdata2 <- LFQData$new(istar2$data, istar2$config)
lfqdata2 <- lfqdata2$get_Transformer()$intensity_array(log2)$lfq
head(lfqdata2$hierarchy())
internal <- lfqdata2$get_subset(head(lfqdata2$hierarchy()))
internal$hierarchy()
tr <- lfqdata2$get_Transformer()
tr$center_to_reference(internal)
pl <- tr$lfq$get_Plotter()
pl$intensity_distribution_density()
lfqdata2$get_Plotter()$intensity_distribution_density()
robscale <- lfqdata2$get_Transformer()$robscale_subset(internal)$lfq
robscale$get_Plotter()$intensity_distribution_density()

```

lfq_power_t_test_proteins

Compute theoretical sample sizes from factor level standard deviations

Description

Compute theoretical sample sizes from factor level standard deviations

Usage

```

lfq_power_t_test_proteins(
  stats_res,
  delta = c(0.59, 1, 2),
  power = 0.8,
  sig.level = 0.05,
  min.n = 1.5
)

```

Arguments

stats_res	data.frame ‘summarize_stats’ output
delta	effect size you are interested in
power	of test
sig.level	P-Value
min.n	smallest n to determine

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()
ldata <- LFQData$new(bb1$data, bb1$config)
stats_res <- summarize_stats(ldata)
bb <- lfq_power_t_test_proteins(stats_res)
```

lfq_power_t_test_quantiles_V2
estimate sample sizes

Description

estimate sample sizes

Usage

```
lfq_power_t_test_quantiles_V2(
  quantile_sd,
  delta = c(0.59, 1, 2),
  power = 0.8,
  sig.level = 0.05,
  min.n = 1.5
)
```

Arguments

quantile_sd	output of ‘summarize_stats_quantiles’
delta	effect size you are interested in
power	of test
sig.level	P-Value
min.n	smallest n to determine

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb1$data, bb1$config)
stats_res <- summarize_stats(lfq)
xx <- summarize_stats_quantiles(stats_res, lfq$relevant_factor_keys(), probs = c(0.5, 0.8))
bbb <- lfq_power_t_test_quantiles_V2(xx$long)
bbb <- dplyr::bind_rows(bbb)
summary <- bbb |>
  dplyr::select( -N_exact, -quantiles, -sdtrimmed ) |>
  tidyr::pivot_wider(names_from = delta, values_from = N)
```

linfct_from_model	<i>get linfct from model</i>
-------------------	------------------------------

Description

get linfct from model

Usage

```
linfct_from_model(m, as_list = TRUE)
```

Arguments

m linear model

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpFacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimp](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limp](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#),

```
impute_refit_singular(), linfct_matrix_contrasts(), list_facades(), lookup_facade(),
merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_
moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(),
sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(),
sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(),
summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()
```

Examples

```
m <- sim_make_model_lm("factors")
linfct <- linfct_from_model(m, as_list = TRUE)

linfct$linfct_factors
linfct$linfct_interactions
lf <- matrix(
c(1, 1, 1, 1, 0.5, 0.5, 0, 1, 0, 1, 0.5, 0.5),
nrow = 4,
byrow = FALSE,
dimnames = list(c("BackgroundX", "BackgroundZ", "TreatmentA", "TreatmentB"),
c("(Intercept)", "TreatmentB", "BackgroundZ"))
)
stopifnot(lf == linfct$linfct_factors)
m <- sim_make_model_lm("interaction")
linfct <- linfct_from_model(m)

m <- lm(Petal.Width ~ Species, data = iris)
linfct_from_model(m)
xx <- data.frame( Y = 1:10 , Condition = c(rep("a",5), rep("b",5)) )
m <- lm(Y ~ Condition, data = xx)
linfct_from_model(m)
xx <- data.frame( Y = 1:10 , Condition = c(rep("a",5), rep("b.b",5)) )
m <- lm(Y ~ Condition, data = xx)
linfct_from_model(m)
xx <- data.frame( Y = 1:10 , Condition = c(rep("a",5), rep("ab",5)) )
m <- lm(Y ~ Condition, data = xx)
linfct_from_model(m)
```

linfct_matrix_contrasts

linfct_matrix_contrasts

Description

linfct_matrix_contrasts

Usage

```
linfct_matrix_contrasts(linfct, contrasts, p.message = FALSE)
```

Arguments

linfct	linear functions as created by linfct_from_model
contrasts	named character vector of contrasts to determine linear functions for
p.message	print messages default FALSE

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_result\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
m <- sim_make_model_lm( "factors")
Contr <- c("TreatmentA_vs_B" = "TreatmentA - TreatmentB",
          "BackgroundX_vs_Z" = "BackgroundX - BackgroundZ",
          "IntoflintoA" = "`TreatmentA:BackgroundX` - `TreatmentA:BackgroundZ`",
          "IntoflintoB" = "`TreatmentB:BackgroundX` - `TreatmentB:BackgroundZ`",
          "IntoflintoX" = "`TreatmentA:BackgroundX` - `TreatmentB:BackgroundX`",
          "IntoflintoZ" = "`TreatmentA:BackgroundZ` - `TreatmentB:BackgroundZ`",
          "interactXZ" = "IntoflintoX - IntoflintoZ",
          "interactAB" = "IntoflintoA - IntoflintoB"
        )
linfct <- linfct_from_model(m, as_list = FALSE)
x <- linfct_matrix_contrasts(linfct, Contr )
stopifnot(sum(x["interactXZ",]) == 0 )
stopifnot(sum(x["interactAB",]) == 0 )

m <- sim_make_model_lm( "interaction")
linfct <- linfct_from_model(m, as_list = FALSE)
x <- linfct_matrix_contrasts(linfct, Contr )
stopifnot(sum(x["interactXZ",]) == 1 )
stopifnot(sum(x["interactAB",]) == 1 )
```

list_facades

List currently registered contrast facades

Description

List currently registered contrast facades

Usage

```
list_facades()
```

Value

A list mirroring [FACADE_REGISTRY](#) but reflecting any entries added via [register_facade](#) from downstream packages.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_limma\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
head(names(list_facades()))
```

```
list_to_AnalysisConfiguration
```

read minimal yaml to reconstruct configuration

Description

read minimal yaml to reconstruct configuration

Usage

```
list_to_AnalysisConfiguration(dd)
```

Arguments

dd list with table and parameter elements as produced by [R6_extract_values](#)

Value

The computed result.

Examples

```
DEAconfig <- AnalysisConfiguration$new()
DEAconfig$file_name <- "Replicate.Name"
DEAconfig$hierarchy[["protein_Id"]] <- "Protein.Name"
DEAconfig$hierarchy[["peptide_Id"]] <- "Peptide.Sequence"
DEAconfig$hierarchy[["precursor_Id"]] <- c("Peptide.Sequence", "Precursor.Charge")
DEAconfig$hierarchy[["fragment_Id"]] <- c(
  "Peptide.Sequence", "Precursor.Charge",
  "Fragment.Ion", "Product.Charge"
)
DEAconfig$ident_q_value <- "annotation_QValue"
DEAconfig$set_response("Area")
DEAconfig$isotope_label <- "Isotope.Label"
configList <- prolfqua::R6_extract_values(DEAconfig)
stopifnot(class(configList) == "list")
config <- list_to_AnalysisConfiguration(configList)
all.equal(prolfqua::R6_extract_values(config), configList)
```

list_to_ContrastConfiguration

Rebuild a ContrastConfiguration from a list

Description

Round-trip partner of [R6_extract_values](#), which keeps the fields but drops the methods. Use it to restore the column-role mapping from a serialized result artifact (e.g. ‘SummarizedExperiment’ metadata or an AnnData ‘uns’ entry) so consumers can call `has_pvalue()` and friends again.

Usage

```
list_to_ContrastConfiguration(dd)
```

Arguments

dd named list of [ContrastConfiguration](#) fields

Value

A [ContrastConfiguration](#).

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_config\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
cfg <- ContrastConfiguration$new(
  subject_id = "protein_Id",
  contrast_col = "Bait",
  pvalue_col = NA_character_
)
values <- R6_extract_values(cfg)
restored <- list_to_ContrastConfiguration(values)
stopifnot(all.equal(R6_extract_values(restored), values))
restored$has_pvalue()
```

lookup_facade	<i>Look up a contrast facade by short name</i>
---------------	--

Description

Look up a contrast facade by short name

Usage

```
lookup_facade(name)
```

Arguments

name short method name (e.g. "lm", "saint").

Value

Registry entry list with at least class, needs, package, needs_saint_annotation; or NULL if no entry exists.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov_rfit_prolfqua\(\)](#)

Examples

```
lookup_facade("lm")$class
is.null(lookup_facade("does-not-exist"))
```

LR_test	<i>Likelihood ratio test</i>
---------	------------------------------

Description

Likelihood ratio test

Usage

```
LR_test(
  complete_models,
  model_name,
  complete_models_int,
  model_name_int,
  subject_id = "protein_Id",
  path = NULL
)
```

Arguments

- complete_models table with models (see build model)
- model_name name of model
- complete_models_int reduced model
- model_name_int name of reduced model
- subject_id subject id typically Accession or protein_Id
- path default NULL, set to a directory if you need to write diagnostic plots.

Value

The computed result.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpaf](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpaf](#), [build_model_logistf](#)

```

compute_borrowed_variance(), compute_borrowed_variance_limma(), compute_contrast(),
compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(),
get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(),
linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_res,
model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(),
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()

```

Examples

```

data_2Factor <- prolfqua::sim_lfq_data_2factor_config(
  Nprot = 200,
  with_missing = TRUE,
  weight_missing = 2)

pMerged <- LFQData$new(data_2Factor$data, data_2Factor$config)

pMerged$response()
pMerged$factors()

formula_condition_and_Batches <-
  prolfqua::strategy_lm("abundance ~ Treatment + Background")
modCB <- prolfqua::build_model(
  pMerged,
  formula_condition_and_Batches)

formula_condition <-
  prolfqua::strategy_lm("abundance ~ Treatment")
modC <- prolfqua::build_model(
  pMerged,
  formula_condition)

tmp <- LR_test(modCB$model_df, "modCB", modC$model_df, "modB")
hist(tmp$likelihood_ratio_test.pValue)

```

main_effect_contrasts *main effects contrasts*

Description

main effects contrasts

Usage

```
main_effect_contrasts(primary_levels, secondary_levels)
```

Arguments

```

primary_levels  character vector of primary factor levels
secondary_levels
                 character vector of secondary factor levels

```

Value

Contrast definitions or contrast results.

See Also

Other contrasts: [annotation_add_contrasts\(\)](#), [generate_contrasts\(\)](#), [generate_contrasts_for_factor\(\)](#), [interaction_contrasts\(\)](#), [level_specific_contrasts\(\)](#)

Examples

```
primary_levels <- c("MI", "MINOCAM")
secondary_levels <- c("T150", "T0", "T300")
main_effect_contrasts(primary_levels, secondary_levels)
main_effect_contrasts(secondary_levels, primary_levels)
```

make_interaction_column

create interaction column from factors

Description

create interaction column from factors

Usage

```
make_interaction_column(data, columns, sep = ".")
```

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_reduced_hierarchy_conf](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
xx <- data.frame(A = c("a", "a", "a"), B = c("d", "d", "e"))
x <- make_interaction_column(xx, c("B", "A"))
x <- make_interaction_column(xx, c("A"))
bb <- prolfqua::sim_lfq_data_protein_config()
config <- bb$config
analysis <- bb$data

config$factor_depth <- 1
make_interaction_column(analysis,
  config$factor_keys_depth())
```

```
make_reduced_hierarchy_config
```

Make reduced hierarchy configuration

Description

Make reduced hierarchy configuration

Usage

```
make_reduced_hierarchy_config(config, work_intensity, hierarchy)
```

Arguments

config	AnalysisConfiguration
work_intensity	work intensity column
hierarchy	new reduced hierarchy

Value

AnalysisConfiguration with reduced hierarchy

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
bb <- prolfqua::sim_lfq_data_peptide_config()

red <- make_reduced_hierarchy_config(bb$config,
  "testintensity",
  bb$config$hierarchy[1])
stopifnot(red$get_response() == "testintensity")
stopifnot(length(red$hierarchy) == 1)
```

matrix_to_tibble	<i>matrix or data.frame to tibble (taken from tidyquant)</i>
------------------	--

Description

matrix or data.frame to tibble (taken from tidyquant)

Usage

```
matrix_to_tibble(x, preserve_row_names = "row.names", ...)
```

Arguments

`x` a matrix
`preserve_row_names` give name to rownames column, if NULL discard rownames
`...` further parameters passed to `as_tibble`

Value

The computed result.

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

Examples

```
x <- matrix(rnorm(20), ncol=4)
rownames(x) <- LETTERS[seq_len(nrow(x))]
matrix_to_tibble(x)
!(is.matrix(x) || is.data.frame(x))
```

```
merge_contrasts_results
```

Merge contrast results coming from two different model.

Description

Typically used to fill proteins missing from [Contrasts](#) with estimates from another model, e.g. the LOD-imputed refits of [ContrastsLMImputeFacade](#)

Usage

```
merge_contrasts_results(prefer, add, model_name = "mergedModel")
```

Arguments

`prefer` contrasts to use preferentially
`add` contrasts to add from if missing in prefer
`model_name` name of the merged model default "mergedModel"

Value

Contrast definitions or contrast results.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
prefer_df <- data.frame(
  protein_Id = c("P1", "P2"),
  contrast = "A_vs_B",
  modelName = "prefer",
  diff = c(1, 2),
  p.value = c(0.01, 0.2),
  FDR = c(0.02, 0.2),
  statistic = c(3, NA)
)
add_df <- data.frame(
  protein_Id = c("P1", "P2"),
  contrast = "A_vs_B",
  modelName = "add",
  diff = c(1.1, 2.1),
  p.value = c(0.02, 0.03),
  FDR = c(0.03, 0.04),
  statistic = c(2.8, 2.5)
)
prefer <- ContrastsTable$new(prefer_df, subject_id = "protein_Id", model_name = "prefer")
add <- ContrastsTable$new(add_df, subject_id = "protein_Id", model_name = "add")
merged <- merge_contrasts_results(prefer, add)
merged$merged$get_contrasts()
```

missigness_histogram *Histogram summarizing missingness*

Description

Histogram summarizing missingness

Usage

```
missigness_histogram(
  lfqdata,
  showempty = FALSE,
  factors = lfqdata$relevant_factor_keys(),
  alpha = 0.1
)
```

Arguments

lfqdata	LFQData object
showempty	show empty values
factors	factor columns to use
alpha	transparency

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Other imputation: [get_contrast\(\)](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
pl <- missigness_histogram(lfq, showempty = FALSE)
stopifnot("ggplot" %in% class(pl))
pl <- missigness_histogram(lfq, showempty = TRUE)
stopifnot("ggplot" %in% class(pl))
```

MissingHelpers

Group statistics and limit of detection

Description

Group statistics and limit of detection

Group statistics and limit of detection

Details

Computes per-group statistics of the abundances and, from them, the limit of detection (LOD) used by [build_model_impute](#) to impute groups without observations.

Value

An R6 class generator.

Public fields

data data
 config config
 prob quantile of groups with one observed value to estimate LOD
 stats data.frame with group statistics

Methods**Public methods:**

- [MissingHelpers\\$new\(\)](#)
- [MissingHelpers\\$get_stats\(\)](#)
- [MissingHelpers\\$get_lod\(\)](#)
- [MissingHelpers\\$clone\(\)](#)

Method new(): initialize*Usage:*

MissingHelpers\$new(data, config, prob = 0.5)

Arguments:

data data
 config config
 prob default 0.5, median of groups with one observed value

Method get_stats(): get data.frame with statistics*Usage:*

MissingHelpers\$get_stats()

Returns: data.frame

Method get_lod(): determine limit of detection computes quantile of abundances in groups with a single observation; without such groups, of the partly observed groups with the fewest observations, and when every group is complete or empty, the lowest group mean

Usage:

MissingHelpers\$get_lod()

Returns: numeric LOD**Method** clone(): The objects of this class are cloneable with this method.*Usage:*

MissingHelpers\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Examples

```
dd <- prolfqua::sim_lfq_data_protein_config(Nprot = 100, weight_missing = 2)
mh <- prolfqua::MissingHelpers$new(dd$data, dd$config, prob = 0.8)
xx <- mh$get_stats()
xx <- mh$get_lod()

dd <- prolfqua::sim_lfq_data_2factor_config(Nprot = 100, weight_missing = 0.1)
mh <- prolfqua::MissingHelpers$new(dd$data, dd$config, prob = 0.8)
mh$get_stats()$interaction |> table()
mh$get_lod()
```

missingness_per_condition

Summarize missing in condition as barplot

Description

Summarize missing in condition as barplot

Usage

```
missingness_per_condition(lfqdata, factors = lfqdata$relevant_factor_keys())
```

Arguments

lfqdata	LFQData object
factors	factor columns to use

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Other imputation: [get_contrast\(\)](#), [missigness_histogram\(\)](#), [missingness_per_condition_cumsum\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
res <- missingness_per_condition(lfq)
stopifnot("ggplot" %in% class(res$figure))
stopifnot(ncol(res$data) >= 5)
```

missingness_per_condition_cumsum

cumulative sums of missing

Description

cumulative sums of missing

Usage

```
missingness_per_condition_cumsum(
  lfqdata,
  factors = lfqdata$relevant_factor_keys()
)
```

Arguments

lfqdata	LFQData object
factors	factor columns to use

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_b](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Other imputation: [get_contrast\(\)](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
res <- missingness_per_condition_cumsum(lfq)
stopifnot("ggplot" %in% class(res$figure))
stopifnot(ncol(res$data) >= 6)
```

Model	<i>R6 class representing modelling result</i>
-------	---

Description

R6 class representing modelling result

R6 class representing modelling result

Value

An R6 class generator.

Super class

```
prolfqua::ModelInterface -> Model
```

Public fields

model_df data.frame with modelling data and model.

model_name name of model

subject_id e.g. protein_Id

model_strategy function to create the models

p.adjust function to adjust p-values

Methods**Public methods:**

- [Model\\$new\(\)](#)
- [Model\\$get_coefficients\(\)](#)
- [Model\\$get_anova\(\)](#)
- [Model\\$clone\(\)](#)

Method `new()`: initialize*Usage:*

```
Model$new(
  model_df,
  model_strategy,
  model_name,
  subject_id = "protein_Id",
  p.adjust = prolfqua::adjust_p_values
)
```

Arguments:

`model_df` dataframe with modelling results
`model_strategy` model_strategy see [strategy_lmer](#)
`model_name` name of model
`subject_id` subject column name
`p.adjust` method to adjust p-values

Method `get_coefficients()`: return model coefficient table*Usage:*

```
Model$get_coefficients()
```

Method `get_anova()`: return anova table*Usage:*

```
Model$get_anova()
```

Method `clone()`: The objects of this class are cloneable with this method.*Usage:*

```
Model$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#),

```

build_model_glm_protein(), build_model_impute(), build_model_limma(), build_model_limma_impute(),
build_model_limma_voom(), build_model_limma_voom_impute(), build_model_limpa(), build_model_logistf(),
compute_borrowed_variance(), compute_borrowed_variance_limma(), compute_contrast(),
compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(),
get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(),
linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_res,
model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(),
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()

```

Examples

```

istar <- sim_lfq_data_peptide_config(Nprot = 20)
lfqdata <- LFQData$new(istar$data, istar$config)
lfqdata <- lfqdata$get_Transformer()$log2()$lfq
model_name <- "f_condition_r_peptide"
formula_randomPeptide <-
  strategy_lmer(paste0(lfqdata$response(), " ~ group_ + (1 | peptide_Id)"),
    model_name = model_name)
mod <- prolfqua::build_model(
  lfqdata,
  formula_randomPeptide,
  model_name = model_name)

mod$model_df
aovtable <- mod$get_anova()
mod$get_coefficients()
mod$coef_histogram()
mod$coef_volcano()
mod$coef_pairs()
mod$anova_histogram()

```

ModelFirth

R6 class representing modelling result

Description

R6 class representing modelling result

R6 class representing modelling result

Value

An R6 class generator.

Super class

`prolfqua::ModelInterface` -> ModelFirth

Public fields

models data.frame with modelling data and model.
 model_name name of model
 subject_id e.g. protein_Id
 p.adjust function to adjust p-values

Methods**Public methods:**

- `ModelFirth$new()`
- `ModelFirth$get_coefficients()`
- `ModelFirth$get_anova()`
- `ModelFirth$anova_histogram()`
- `ModelFirth$clone()`

Method `new()`: initialize*Usage:*

```
ModelFirth$new(
  models,
  model_name = "modelFirth",
  subject_id = "protein_Id",
  p.adjust = p.adjust::adjust_p_values
)
```

Arguments:

models dataframe with modelling results
 model_name name of model
 subject_id subject column name
 p.adjust method to adjust p-values

Method `get_coefficients()`: return model coefficient table*Usage:*

```
ModelFirth$get_coefficients()
```

Method `get_anova()`: return anova table*Usage:*

```
ModelFirth$get_anova()
```

Method `anova_histogram()`: histogram of ANOVA results*Usage:*

```
ModelFirth$anova_histogram(what = c("p.value", "FDR"))
```

Arguments:

what show either "Pr..F." or "FDR.Pr..F."

Method `clone()`: The objects of this class are cloneable with this method.*Usage:*

```
ModelFirth$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [Modellimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimp](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limp\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limp\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config(Nprot = 10, with_missing = TRUE,
  weight_missing = 0.5, seed = 3)
istar$data <- prolfqua::encode_bin_resp(LFQData$new(istar$data, istar$config))
istar$config$bin_resp <- "bin_resp"
tmp <- LFQData$new(istar$data, istar$config)
formula <- paste0(tmp$get_config()$bin_resp , "~ group_")
mod <- build_model_logistf(tmp, formula)
tmp <- mod$get_coefficients()
mod$coef_histogram()
mod$coef_pairs()
mod$get_anova()
mod$coef_volcano()
mod$anova_histogram()

istar <- prolfqua::sim_lfq_data_protein_config(Nprot = 10, with_missing = TRUE,
  weight_missing = 0.5, seed = 3)
istar$data <- prolfqua::encode_bin_resp(LFQData$new(istar$data, istar$config))
istar$config$bin_resp <- "bin_resp"
tmp <- LFQData$new(istar$data, istar$config)
formula <- paste0(tmp$get_config()$bin_resp , "~ group_")
mod <- build_model_logistf(tmp, formula)
tmp <- mod$get_coefficients()
stopifnot(nrow(tmp) == 30)
mod$coef_histogram()
mod$coef_pairs()
mod$get_anova()
mod$coef_volcano()
mod$anova_histogram()
```

ModelInterface*R6 interface class representing modelling result*

Description

R6 interface class representing modelling result

R6 interface class representing modelling result

Value

An R6 class generator.

Methods**Public methods:**

- `ModelInterface$get_coefficients()`
- `ModelInterface$get_anova()`
- `ModelInterface$coef_histogram()`
- `ModelInterface$coef_volcano()`
- `ModelInterface$coef_pairs()`
- `ModelInterface$anova_histogram()`
- `ModelInterface$clone()`

Method `get_coefficients()`: return model coefficients

Usage:

`ModelInterface$get_coefficients()`

Returns: data.frame

Method `get_anova()`: perform ANOVA analysis

Usage:

`ModelInterface$get_anova()`

Returns: data.frame

Method `coef_histogram()`: histogram of coefficient p-values

Usage:

`ModelInterface$coef_histogram()`

Returns: list with plot (ggplot) and file name

Method `coef_volcano()`: volcano plot of non-intercept coefficients

Usage:

`ModelInterface$coef_volcano()`

Returns: list with plot (ggplot) and file name

Method `coef_pairs()`: coefficient estimates in wide format, one column per coefficient, for a pairs plot

Usage:

`ModelInterface$coef_pairs()`

Returns: list with plot (data.frame) and file name

Method `anova_histogram()`: histogram of ANOVA p-values or FDR

Usage:

```
ModelInterface$anova_histogram(what = c("p.value", "FDR"))
```

Arguments:

`what` show either "p.value" or "FDR"

Returns: list with plot (ggplot) and file name

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ModelInterface$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
mi <- ModelInterface$new()

testthat::expect_error(mi$get_coefficients())
testthat::expect_error(mi$get_anova())
testthat::expect_error(mi$coef_histogram())
testthat::expect_error(mi$coef_volcano())
testthat::expect_error(mi$coef_pairs())
testthat::expect_error(mi$anova_histogram())
```

ModelLimma

R6 class representing a limma modelling result

Description

R6 class representing a limma modelling result

R6 class representing a limma modelling result

Details

Same API as [Model](#): `get_anova()`, `get_coefficients()`, `coef_histogram()`, `coef_volcano()`, `anova_histogram()`.

Value

An R6 class generator.

Super class

`prolfqua::ModelInterface` -> ModelLimma

Public fields

`fit` limma MArrayLM object from `lmFit`
`subject_id` protein ID column name(s)
`model_name` model name
`rowdata` protein ID mapping from `to_wide()`\$rowdata
`trend` passed to `eBayes`
`robust` passed to `eBayes`
`dummy_model` one fitted `lm` for `linfct` extraction
`p.adjust` function to adjust p-values
`imputed_proteins` character vector of protein ids (matching `rownames(fit$coefficients)`) that were rescued via LOD imputation. `ContrastsLimma$get_contrasts()` reads this to tag refit rows with a `"_imputed"` modelName suffix so the rescue is visible downstream. Empty for builders that do not impute.

Methods**Public methods:**

- `ModelLimma$new()`
- `ModelLimma$get_coefficients()`
- `ModelLimma$get_anova()`
- `ModelLimma$clone()`

Method `new()`: initialize `ModelLimma`

Usage:

```

ModelLimma$new(
  fit,
  subject_id,
  model_name,
  rowdata,
  trend = FALSE,
  robust = FALSE,
  dummy_model = NULL,
  p.adjust = p.adjust,
  imputed_proteins = character(0)
)

```

Arguments:

`fit` limma MArrayLM from `lmFit`
`subject_id` protein ID column name(s)
`model_name` model name
`rowdata` protein ID mapping
`trend` passed to `eBayes`
`robust` passed to `eBayes`
`dummy_model` one fitted `lm` for `linfct` extraction
`p.adjust` function to adjust p-values
`imputed_proteins` character vector of LOD-rescued protein ids

Method `get_coefficients()`: return model coefficient table in long format

Usage:

```
ModelLimma$get_coefficients()
```

Returns: data.frame

Method `get_anova()`: return anova table (F-test per protein across all non-intercept coefficients)

Usage:

```
ModelLimma$get_anova()
```

Returns: data.frame

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ModelLimma$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(Nprot = 50)
lProt <- LFQData$new(istar$data, istar$config)
lProt$rename_response("transformedIntensity")
strat <- strategy_limma("transformedIntensity ~ group_")
mod <- build_model_limma(lProt, strat)

coeffs <- mod$get_coefficients()
head(coeffs)
anova_tbl <- mod$get_anova()
head(anova_tbl)
mod$coef_histogram()
mod$coef_volcano()
```

```
mod$anova_histogram()
```

model_analyse	<i>analyses lmer4 and lm models created using help function ‘strategy_lm’ or ‘strategy_lmer’</i>
---------------	--

Description

used in project p2901

Usage

```
model_analyse(
  pepIntensity,
  model_strategy,
  subject_id = "protein_Id",
  model_name = "Model",
  label = model_name,
  progress = getOption("prolfqua.progress", NULL)
)
```

Arguments

label	short label naming this fitting pass in progress output; defaults to ‘model_name’.
progress	progress reporter selector passed to [.make_progress()]; ‘NULL’ (default) keeps the legacy terminal ‘progress::progress_bar’, a ‘function(i, total, label)’ or ‘“message”‘; emits batch-visible progress. Defaults to ‘getOption(“prolfqua.progress”, NULL)’.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimp](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limp\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limp\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
x <- sim_lfq_data_peptide_config()
formula_randomPeptide <-
  strategy_lmer("abundance ~ group_ + (1 | peptide_Id)")
mr <- model_analyse( x$data,
  formula_randomPeptide,
  subject_id = x$config$hierarchy_keys_depth())
stopifnot(nrow(get_complete_model_fit(mr$model_df)) == 6)
```

model_summary

Summarize modelling and error reporting

Description

Summarize modelling and error reporting

Usage

```
model_summary(mod)
```

Arguments

mod model table see [build_model](#)

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_voom\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
D <- prolfqua::sim_lfq_data_peptide_config(Nprot = 20, weight_missing = 0.1)
formula_rp <- strategy_lmer("abundance ~ group_ + (1 | peptide_Id) + (1 | sampleName)")
mod <- prolfqua::build_model(
  LFQData$new(D$data, D$config), formula_rp)
res <- model_summary(mod)
stopifnot(is.list(res))
stopifnot(all(c("exists", "isSingular") %in% names(res)))
```

moderated_p_deqms	<i>DEqMS-style count-dependent variance moderation</i>
-------------------	--

Description

Applies count-dependent empirical Bayes variance shrinkage to a contrast result table. Proteins quantified from many peptides get less shrinkage; proteins from few peptides get more.

Usage

```
moderated_p_deqms(
  mm,
  count_col,
  df = "df",
  estimate = "diff",
  loess_span = 0.75,
  confint = 0.95
)
```

Arguments

mm	data.frame from one contrast group with columns: sigma, df, statistic, std.error, and the estimate column
count_col	name of column with peptide/PSM count per protein
df	name of the degrees of freedom column
estimate	name of the fold change column
loess_span	span parameter for LOESS fit (default 0.75)
confint	confidence level for intervals (default 0.95)

Value

data.frame with added moderated.* columns

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpute](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNested](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#),

```

ContrastsRfitImputeFacade, ContrastsTable, INTERNAL_FUNCTIONS_BY_FAMILY, LR_test(),
Model, ModelFirth, ModelLimma, StrategyLM, StrategyLimma, StrategyLimpa, StrategyLmer,
StrategyLogistf, StrategyRLM, StrategyRfit, build_contrast_analysis(), build_model(),
build_model_glm_peptide(), build_model_glm_protein(), build_model_impute(), build_model_limma(),
build_model_limma_impute(), build_model_limma_voom(), build_model_limma_voom_impute(),
build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance_
compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(),
get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(),
impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(),
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms_long
moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(),
sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(),
sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(),
summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

```

Examples

```

mm <- data.frame(
  sigma = c(0.25, 0.32, 0.28, 0.40, 0.35),
  df = rep(6, 5),
  statistic = c(2.1, -1.8, 0.5, 3.0, -2.2),
  std.error = c(0.8 / 2.1, 0.6 / 1.8, 0.2 / 0.5, 1.2 / 3.0, 0.9 / 2.2),
  diff = c(0.8, -0.6, 0.2, 1.2, -0.9),
  count = c(2, 3, 4, 6, 8)
)
res <- moderated_p_deqms(mm, count_col = "count")
"moderated.p.value" %in% colnames(res)

```

moderated_p_deqms_long

DEqMS-style moderation for long contrast table

Description

Splits by contrast group and applies [moderated_p_deqms](#) to each group independently.

Usage

```

moderated_p_deqms_long(
  mm,
  count_col,
  group_by_col = "contrast",
  estimate = "diff",
  loess_span = 0.75
)

```

Arguments

mm	contrast result data.frame (long format)
count_col	name of column with peptide/PSM count per protein
group_by_col	column to group by (default "contrast")
estimate	name of the fold change column
loess_span	span parameter for LOESS fit (default 0.75)

Value

data.frame with moderated columns added

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
mm <- data.frame(
  contrast = rep(c("A_vs_B", "C_vs_D"), each = 5),
  sigma = rep(c(0.25, 0.32, 0.28, 0.40, 0.35), 2),
  df = rep(6, 10),
  statistic = rep(c(2.1, -1.8, 0.5, 3.0, -2.2), 2),
  std.error = rep(c(0.8 / 2.1, 0.6 / 1.8, 0.2 / 0.5, 1.2 / 3.0, 0.9 / 2.2), 2),
  diff = rep(c(0.8, -0.6, 0.2, 1.2, -0.9), 2),
  count = rep(c(2, 3, 4, 6, 8), 2)
)
res <- moderated_p_deqms_long(mm, count_col = "count")
table(res$contrast)
```

moderated_p_limma

Moderate p-values - limma approach

Description

Moderate p-values - limma approach

Usage

```
moderated_p_limma(
  contrast_df,
  df = "df",
```

```

estimate = "diff",
robust = FALSE,
confint = 0.95,
variance_floor = NULL
)

```

Arguments

`variance_floor` optional positive lower bound for posterior variances

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```

contrast_df <- data.frame(
  sigma = c(0.25, 0.32, 0.28, 0.40, 0.35),
  df = rep(6, 5),
  statistic = c(2.1, -1.8, 0.5, 3.0, -2.2),
  std.error = c(0.8 / 2.1, 0.6 / 1.8, 0.2 / 0.5, 1.2 / 3.0, 0.9 / 2.2),
  diff = c(0.8, -0.6, 0.2, 1.2, -0.9)
)
res <- moderated_p_limma(contrast_df)
"moderated.p.value" %in% colnames(res)

```

multigroup_volcano

plot volcano given multiple contrasts

Description

plot volcano given multiple contrasts

Usage

```
multigroup_volcano(
  .data,
  effect = "fc",
  significance = "p.adjust",
  contrast = "condition",
  colour = "colour",
  xintercept = c(-2, 2),
  yintercept = 0.05,
  label = NULL,
  size = 1,
  segment.size = 0.3,
  segment.alpha = 0.3,
  scales = "fixed",
  max_nr_significant_text = 20
)
```

Arguments

.data	data in long format
effect	column containing effect sizes
significance	column containing p-values, q.values etc
contrast	column with contrast
colour	colouring of points
xintercept	fc thresholds
yintercept	significance threshold
label	column containing labels
size	controls size of text
segment.size	controls size of lines
segment.alpha	controls visibility of lines
scales	parameter to <code>ggplot2::facet_wrap</code>
max_nr_significant_text	maximum number of significant labels to display

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

Examples

```
show <- data.frame(logFC = rnorm(100, 0, 1), adj.P.Val = runif(100, 0, 1),
  Condition = sample(c("a","b")), colour = "forward", Name = paste0("n", 1:100))
prolfqua::multigroup_volcano( show,
  effect="logFC",
  significance = "adj.P.Val",
  contrast="Condition",
  colour=NULL,
  label="Name",
  max_nr_significant_text = 300)
```

new_imputed_model	<i>Create an imputed model wrapper</i>
-------------------	--

Description

Wraps a refitted model (lm, rfit, ...) with borrowed covariance information. S3 generics vcov(), sigma(), and df.residual() dispatch to the borrowed values instead of the original model's, while coef()/terms() still fall through to the wrapped fit. The wrapper is backend-neutral: the class is prepended to whatever the refit already carries (e.g. c("imputed_model", "lm") or c("imputed_model", "rfit_prolfqua", "rfit")).

Usage

```
new_imputed_model(
  model,
  borrowed_vcov,
  borrowed_sigma,
  borrowed_df,
  n_observed
)
```

Arguments

model	model object fitted on imputed data
borrowed_vcov	matrix, borrowed variance-covariance matrix
borrowed_sigma	numeric, borrowed residual standard error / scale
borrowed_df	numeric, borrowed residual degrees of freedom
n_observed	integer, number of non-imputed observations

Value

imputed_model object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#),

```
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), pivot_model_contrasts_to_wide(), register_facade(),
sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(),
sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(),
summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()
```

Examples

```
# Fit a normal lm, then wrap it with borrowed covariance
dat <- data.frame(group_ = rep(c("A", "B"), each = 4),
                  y = c(20.1, 20.5, 19.8, 20.3, 22.1, 22.4, 21.9, 22.2))
fit <- lm(y ~ group_, data = dat)

# Wrap with borrowed variance (in practice these come from donor pool)
wrapped <- prolfqua::new_imputed_model(fit,
  borrowed_vcov = vcov(fit),
  borrowed_sigma = 0.8,
  borrowed_df = 6,
  n_observed = 5)

# S3 dispatch returns borrowed values
stopifnot(inherits(wrapped, "imputed_model"))
stopifnot(sigma(wrapped) == 0.8)
stopifnot(df.residual(wrapped) == 6)
# coefficients() still dispatches to underlying lm
stopifnot(identical(coefficients(wrapped), coefficients(fit)))
```

nr_B_in_A

Count the children of each parent in the hierarchy

Description

The number of distinct ‘level_b’ entries, e.g. peptides, within each ‘level_a’ entry, e.g. protein.

Usage

```
nr_B_in_A(
  pdata,
  level_a,
  level_b,
  name = paste0("nr_", paste(level_b, collapse = "_"), "_IN_", paste(level_a, collapse =
    "_"))
)
```

Arguments

pdata	data.frame
level_a	parent key columns
level_b	child key columns
name	name of the count column

Value

data.frame of the ‘level_a’ keys and the count column

See Also

Other preprocessing: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [apply_to_response_matrix\(\)](#), [filter_proteins_by_peptide\(\)](#), [get_robscales\(\)](#), [robust_scale\(\)](#), [scale_with_subset\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config()
head(prolfqua::nr_B_in_A(istar$data, "protein_Id", "peptide_Id"))
```

nr_children_experiment
<i>Max nr_children across samples per hierarchy unit</i>

Description

Aggregates nr_children per sample (via nr_obs_sample), then takes the max per hierarchy unit.

Usage

```
nr_children_experiment(
  data,
  response,
  hierarchy_keys_depth,
  file_name,
  nr_children_col,
  name_nr_child = "nr_child_exp"
)
```

Arguments

data	data.frame
response	character — intensity column name
hierarchy_keys_depth	character vector — hierarchy columns at current depth
file_name	character — file name column
nr_children_col	character — nr_children column name
name_nr_child	character — output column name

Value

The computed result.

Examples

```
dd <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(dd$data, dd$config)
xd <- nr_children_experiment(lfq$data_long(), lfq$response(),
  lfq$relevant_hierarchy_keys(), lfq$file_name(), lfq$nr_children_col())
stopifnot(min(xd$nr_child_exp) == 1)
```

nr_features_experiment

Count distinct child features per hierarchy unit

Description

Counts the number of distinct child-level entries (e.g. peptides per protein).

Usage

```
nr_features_experiment(
  data,
  hierarchy_keys,
  hierarchy_keys_depth,
  name_nr_child = "nr_child_exp"
)
```

Arguments

`data` `data.frame`

`hierarchy_keys` `character` vector — all hierarchy column names

`hierarchy_keys_depth` `character` vector — hierarchy columns at current depth

`name_nr_child` `character` — output column name

Value

The computed result.

Examples

```
dd <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(dd$data, dd$config)
xd <- nr_features_experiment(lfq$data_long(), lfq$hierarchy_keys(),
  lfq$relevant_hierarchy_keys())
stopifnot(min(xd$nr_child_exp) == 1)
```

nr_obs_sample	<i>Aggregates e.g. protein abundances from peptide abundances</i>
---------------	---

Description

Aggregates e.g. protein abundances from peptide abundances

Usage

```
nr_obs_sample(
  data,
  response,
  hierarchy_keys_depth,
  file_name,
  nr_children_col,
  new_child = nr_children_col
)
```

Arguments

data	data.frame
response	character — intensity column name
hierarchy_keys_depth	character vector — hierarchy columns at current depth
file_name	character — file name column
nr_children_col	character — nr_children column name
new_child	character — output column name

Value

The computed result.

Examples

```
dd <- prolfqua::sim_lfq_data_peptide_config()
dd$data <- na.omit(dd$data)

xd <- nr_obs_sample(na.omit(dd$data), dd$config$get_response(),
  dd$config$hierarchy_keys_depth(), dd$config$file_name, dd$config$nr_children)
xd$nr_children |> table()
# xd |> pivot_wider(id_cols = protein_Id, names_from = sample, values_from = nr_children)

dp <- prolfqua::sim_lfq_data_protein_config()
xp <- nr_obs_sample(dp$data, dp$config$get_response(),
  dp$config$hierarchy_keys_depth(), dp$config$file_name, dp$config$nr_children)
# xp
# xp |> pivot_wider(id_cols = protein_Id, names_from = sample, values_from = nrPeptides)
xp$nrPeptides |> table()
```

PACKAGE_DATA	<i>Package Data</i>
--------------	---------------------

Description

Package Data

Value

A documentation topic.

See Also

Other data: [data_checksummerobust87](#), [data_checksummerobust69](#), [data_ionstar](#), [prolfqua_data\(\)](#), [x5463yzwer453bbb](#)

pairs_smooth	<i>smoothScatter pairs</i>
--------------	----------------------------

Description

smoothScatter pairs

Usage

```
pairs_smooth(data, legend = FALSE, ...)
```

Arguments

data	data matrix or data.frame as normally passed to pairs
legend	add legend to plots
...	params usually passed to pairs

See Also

also [pairs](#)

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [panel_cor\(\)](#), [table_facade\(\)](#)

Examples

```
tmp = matrix(rep((1:100),times = 4) + rnorm(100*4,0,3),ncol=4)
pairs_smooth(tmp,main="small data", legend=TRUE)
pairs_smooth(tmp,main="small data")
pairs_smooth(tmp,log="xy",main="small data", legend=TRUE)
```

panel_cor	<i>correlation panel for pairs plot function (used as default in pairs_smooth)</i>
-----------	--

Description

correlation panel for pairs plot function (used as default in pairs_smooth)

Usage

```
panel_cor(x, y, digits = 2, ...)
```

Arguments

x	numeric data
y	numeric data
digits	number of digits to display
...	not used

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [table_facade\(\)](#)

Examples

```
graphics::plot(1:3, 1:3, type = "n")
panel_cor(1:3, 1:3)
```

pivot_model_contrasts_to_wide	<i>pivot model contrasts matrix to wide format produced by ‘contrasts_linfct’ and ...</i>
-------------------------------	---

Description

pivot model contrasts matrix to wide format produced by ‘contrasts_linfct’ and ...

Usage

```
pivot_model_contrasts_to_wide(
  model_interaction_contrasts,
  subject_id = "protein_Id",
  columns = c("estimate", "p.value", "p.value.adjusted"),
  contrast = "lhs"
)
```

Arguments

model_interaction_contrasts	data.frame with contrast results in long format
subject_id	column name(s) identifying subjects (e.g. protein_Id)
columns	character vector of value columns to pivot wide
contrast	column name containing contrast labels

Value

Contrast definitions or contrast results.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
# this function is used by the contrast classes to implement the to wide method
contrast_df <- data.frame(
  protein_Id = c("P1", "P1", "P2", "P2"),
  lhs = c("A_vs_B", "C_vs_D", "A_vs_B", "C_vs_D"),
  estimate = c(1.0, 0.5, -0.2, 0.7),
  p.value = c(0.01, 0.2, 0.4, 0.05),
  p.value.adjusted = c(0.02, 0.25, 0.4, 0.08)
)
pivot_model_contrasts_to_wide(contrast_df)
```

plot_estimate	<i>Plot feature data and result of aggregation</i>
---------------	--

Description

Plot feature data and result of aggregation

Usage

```
plot_estimate(lfqdata, lfqdata_agg, show.legend = FALSE)
```

Arguments

lfqdata	LFQData object (original data)
lfqdata_agg	LFQData object (aggregated data)
show.legend	logical, show legend in plot

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_s](#), [upset_missing_stats\(\)](#)

Other aggregation: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [aggregate_intensity_top_n\(\)](#), [estimate_intensity\(\)](#), [plot_hierarchies_line\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
lfq <- lfq$get_Transformer()$log2()$lfq
bbMed <- estimate_intensity(lfq, method = "medpolish")
lfq_med <- LFQData$new(bbMed$data, bbMed$config)
tmpMed <- plot_estimate(lfq, lfq_med)
stopifnot("ggplot" %in% class(tmpMed$plots[[1]]))
```

plot_heatmap	<i>plot heatmap with annotations</i>
--------------	--------------------------------------

Description

plot heatmap with annotations

Usage

```
plot_heatmap(
  matrix,
  annotation,
  factor_keys,
  sample_name,
  na_fraction = 0.4,
  show_rownames = FALSE,
  max_rownames_chars = 60,
  max_sample_label_chars = 20,
  ...
)
```

Arguments

<code>matrix</code>	numeric matrix — wide-format intensity data
<code>annotation</code>	<code>data.frame</code> — sample annotation
<code>factor_keys</code>	character vector — factor column names for annotation
<code>sample_name</code>	character — sample name column
<code>na_fraction</code>	maximum fraction of missing values per row. Rows that cannot be meaningfully z-scored are always removed.
<code>show_rownames</code>	if TRUE shows row names, default FALSE
<code>max_rownames_chars</code>	maximum displayed row label length
<code>max_sample_label_chars</code>	maximum displayed sample label length. Labels keep their suffix because sample prefixes are often shared.
<code>...</code>	passed to <code>[ComplexHeatmap::Heatmap()]</code>

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distrib](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_s](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config()
lfq <- LFQData$new(istar$data, istar$config)
wide <- lfq$data_wide(as.matrix = TRUE)
p <- plot_heatmap(wide$data, wide$annotation, lfq$factor_keys(), lfq$sample_name())
stopifnot(methods::is(p, "Heatmap"))
```

plot_heatmap_cor	<i>plot correlation heatmap with annotations</i>
------------------	--

Description

plot correlation heatmap with annotations

Usage

```
plot_heatmap_cor(
  matrix,
  annotation,
  factor_keys,
  sample_name,
  R2 = FALSE,
  color = colorRampPalette(c("white", "red"))(1024),
  max_sample_label_chars = 20,
  ...
)
```

Arguments

matrix	numeric matrix — wide-format intensity data
annotation	data.frame — sample annotation
factor_keys	character vector — factor column names for annotation
sample_name	character — sample name column
R2	logical — plot R-squared instead of correlation
color	color palette
max_sample_label_chars	maximum displayed sample label length. Labels keep their suffix because sample prefixes are often shared.
...	passed to [ComplexHeatmap::Heatmap()]

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_s](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config()
lfq <- LFQData$new(istar$data, istar$config)
wide <- lfq$data_wide(as.matrix = TRUE)
pheat_map <- plot_heatmap_cor(wide$data, wide$annotation,
  lfq$factor_keys(), lfq$sample_name())
stopifnot(methods::is(pheat_map, "Heatmap"))
pheat_map <- plot_heatmap_cor(wide$data, wide$annotation,
```

```
lfq$factor_keys(), lfq$sample_name(), R2 = TRUE)
stopifnot(methods::is(pheat_map, "Heatmap"))
```

```
plot_hierarchies_boxplot
```

plot peptides by factors and it's levels.

Description

plot peptides by factors and it's levels.

Usage

```
plot_hierarchies_boxplot(
  pdata,
  title,
  lfqdata,
  facet_grid_on = NULL,
  beeswarm = TRUE,
  show_mean = TRUE,
  pb
)
```

Arguments

<code>pdata</code>	data.frame
<code>title</code>	name to show
<code>lfqdata</code>	LFQData object
<code>facet_grid_on</code>	on which variable to run <code>facet_grid</code>
<code>beeswarm</code>	use beeswarm default FALSE
<code>show_mean</code>	show mean values
<code>pb</code>	progress bar

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
res <- plot_hierarchies_boxplot_df(lfq$data_long(), lfq)
res$boxplot[[1]]

xnested <- lfq$data_long() |>
  dplyr::group_by(across(all_of(lfq$relevant_hierarchy_keys())) |> tidyr::nest())
p <- plot_hierarchies_boxplot(xnested$data[[1]], xnested$protein_Id[[1]],
  lfq, beeswarm = FALSE, show_mean = TRUE)
p <- plot_hierarchies_boxplot(xnested$data[[1]], xnested$protein_Id[[1]],
  lfq, beeswarm = TRUE)
p
```

plot_hierarchies_boxplot_df

generates peptide level plots for all Proteins

Description

generates peptide level plots for all Proteins

Usage

```
plot_hierarchies_boxplot_df(
  pdata,
  lfqdata,
  hierarchy = lfqdata$relevant_hierarchy_keys(),
  facet_grid_on = NULL
)
```

Arguments

pdata	data.frame
lfqdata	LFQData object
hierarchy	e.g. protein_Id default relevant_hierarchy_keys
facet_grid_on	default NULL

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_s](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
res <- plot_hierarchies_boxplot_df(lfq$data_long(), lfq)
res$boxplot[[1]]

lfq2 <- LFQData$new(
  istar$data |> dplyr::filter(protein_Id %in% sample(protein_Id, 2)),
  istar$config)
res <- plot_hierarchies_boxplot_df(lfq2$data_long(), lfq2)
res$boxplot[[1]]
```

plot_hierarchies_line *Plot peptide intensities of protein as a function of the sample and factor*

Description

Plot peptide intensities of protein as a function of the sample and factor

Usage

```
plot_hierarchies_line(res, protein_name, lfqdata, show.legend = FALSE)
```

Arguments

res	data.frame
protein_name	title of plot
lfqdata	LFQData object
show.legend	logical, show legend in plot

See Also

Other aggregation: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [aggregate_intensity_top_n\(\)](#), [estimate_intensity\(\)](#), [plot_estimate\(\)](#)

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missingness\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
config <- istar$config
analysis <- istar$data

xnested <- analysis |>
  dplyr::group_by(across(all_of(config$hierarchy_keys_depth())) |>
    tidyr::nest())

lfq <- LFQData$new(analysis, config)
prolfqua::plot_hierarchies_line(xnested$data[[1]], xnested$protein_Id[[1]], lfq)
```

```
plot_intensity_distribution_violin
      visualize intensity distributions
```

Description

visualize intensity distributions

visualize intensity distributions

Usage

```
plot_intensity_distribution_violin(
  pdata,
  sample_name,
  response,
  is_transformed = FALSE
)

plot_intensity_distribution_density(
  pdata,
  sample_name,
  response,
  is_transformed = FALSE,
  legend = NA,
  max_legend_samples = 16
)
```

Arguments

<code>pdata</code>	<code>data.frame</code>
<code>sample_name</code>	character — sample column name
<code>response</code>	character — intensity column name
<code>is_transformed</code>	logical — is intensity log-transformed?
<code>legend</code>	show legend. If 'NA', hide automatically when the number of samples is larger than 'max_legend_samples'.
<code>max_legend_samples</code>	maximum number of samples for automatic legend display.

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_s](#), [upset_missing_stats\(\)](#)

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_s](#), [upset_missing_stats\(\)](#)

Examples

```

istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
plot_intensity_distribution_violin(
  lfq$data_long(), lfq$sample_name(), lfq$response(), lfq$is_transformed())

istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
plot_intensity_distribution_density(
  lfq$data_long(), lfq$sample_name(), lfq$response(), lfq$is_transformed())

```

plot_lrt_diagnostics	<i>Write LRT diagnostic plots to PDF</i>
----------------------	--

Description

Writes a histogram of p-values and an empirical CDF to a PDF file. Called by [LR_test](#) when path is non-NULL.

Usage

```
plot_lrt_diagnostics(result, model_name, model_name_int, path)
```

Arguments

result	tibble with a likelihood_ratio_test.pValue column
model_name	name of the full model
model_name_int	name of the reduced/interaction model
path	directory to write the PDF into

plot_na_heatmap	<i>plot heatmap of NA values</i>
-----------------	----------------------------------

Description

plot heatmap of NA values

Usage

```

plot_na_heatmap(
  matrix,
  annotation,
  factor_keys,
  sample_name,
  limitrows = 10000,
  distance = "binary"
)

```

Arguments

matrix	numeric matrix — wide-format intensity data
annotation	data.frame — sample annotation
factor_keys	character vector — factor column names for annotation
sample_name	character — sample name column
limitrows	max rows to display
distance	distance method for clustering

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
wide <- lfq$data_wide(as.matrix = TRUE)
tmp <- plot_na_heatmap(wide$data, wide$annotation, lfq$factor_keys(), lfq$sample_name())
stopifnot(methods::is(tmp, "Heatmap"))
```

plot_pca	<i>plot PCA</i>
----------	-----------------

Description

plot PCA

Usage

```
plot_pca(
  matrix,
  annotation,
  sample_name,
  factor_keys,
  PC = c(1, 2),
  add_txt = FALSE,
  nudge = 0.1
)
```

Arguments

matrix	numeric matrix — wide-format intensity data
annotation	data.frame — sample annotation
sample_name	character — sample name column
factor_keys	character vector — factor column names (first for color, second for shape)
PC	which principal components to plot
add_txt	show sample labels
nudge	label nudge distance

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config(with_missing = TRUE, weight_missing = .8, Nprot = 3000)
lfq <- LFQData$new(istar$data, istar$config)
wide <- lfq$data_wide(as.matrix = TRUE)
tmp <- plot_pca(wide$data, wide$annotation, lfq$sample_name(), lfq$factor_keys(),
  add_txt = TRUE, nudge = 0.01)
stopifnot("ggplot" %in% class(tmp))
tmp <- plot_pca(wide$data, wide$annotation, lfq$sample_name(), lfq$factor_keys())
stopifnot("ggplot" %in% class(tmp))
```

plot_raster	<i>plot heatmap without any clustering (use to show NA's)</i>
-------------	---

Description

plot heatmap without any clustering (use to show NA's)

Usage

```
plot_raster(
  matrix,
  annotation,
  factor_keys,
  sample_name,
  arrange = c("mean", "var"),
  not_na = FALSE,
  show_rownames = FALSE,
  max_rownames_chars = 60,
  max_sample_label_chars = 20,
  ...
)
```

Arguments

matrix	numeric matrix — wide-format intensity data
annotation	data.frame — sample annotation
factor_keys	character vector — factor column names for annotation
sample_name	character — sample name column
arrange	either mean or var
not_na	if true than arrange by nr of NA's first and then by arrange
show_rownames	logical, show row names in heatmap
max_rownames_chars	maximum displayed row label length
max_sample_label_chars	maximum displayed sample label length. Labels keep their suffix because sample prefixes are often shared.
...	additional arguments passed to [ComplexHeatmap::Heatmap()]

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_protein_config()
lfq <- LFQData$new(istar$data, istar$config)
wide <- lfq$data_wide(as.matrix = TRUE)
rs <- plot_raster(wide$data, wide$annotation, lfq$factor_keys(), lfq$sample_name())
stopifnot(methods::is(rs, "Heatmap"))
rs <- plot_raster(wide$data, wide$annotation, lfq$factor_keys(), lfq$sample_name(), "var")
stopifnot(methods::is(rs, "Heatmap"))
```

plot_sample_correlation

visualize correlation among samples

Description

visualize correlation among samples

Usage

```
plot_sample_correlation(matrix)
```

Arguments

matrix	numeric matrix — wide-format intensity data (samples as columns)
--------	--

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missigness_per_condition\(\)](#), [missigness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [upset_interaction_missing_stats\(\)](#), [upset_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
lfq <- lfq$get_Transformer()$log2()$lfq
plot_sample_correlation(lfq$data_wide(as.matrix = TRUE)$data)
```

plot_stat_density	<i>plot density distribution or ecdf of sd, mean or CV</i>
-------------------	--

Description

plot density distribution or ecdf of sd, mean or CV

Usage

```
plot_stat_density(
  pdata,
  factor_key,
  stat = c("CV", "meanAbundance", "sd"),
  ggstat = c("density", "ecdf")
)
```

Arguments

pdata	data.frame with statistics
factor_key	character — factor column name for colouring
stat	sd, mean or CV
ggstat	either density or ecdf

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_mean\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb1$data, bb1$config)
res <- lfq$get_Stats()$stats()
plot_stat_density(res, lfq$factor_keys()[1], stat = "meanAbundance")
plot_stat_density(res, lfq$factor_keys()[1], stat = "sd")
plot_stat_density(res, lfq$factor_keys()[1], stat = "CV")
```

plot_stat_density_median	<i>plot density distribution or ecdf of sd, mean or cv given intensity below and above median</i>
--------------------------	---

Description

plot density distribution or ecdf of sd, mean or cv given intensity below and above median

Usage

```
plot_stat_density_median(  
  pdata,  
  factor_key,  
  stat = c("CV", "meanAbundance", "sd"),  
  ggstat = c("density", "ecdf")  
)
```

Arguments

pdata	data.frame with statistics
factor_key	character — factor column name for faceting
stat	sd, mean or CV
ggstat	either density or ecdf

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()  
lfq <- LFQData$new(bb1$data, bb1$config)  
res <- lfq$get_Stats()$stats()  
plot_stat_density_median(res, lfq$factor_keys()[1], "CV")  
plot_stat_density_median(res, lfq$factor_keys()[1], "meanAbundance")  
plot_stat_density_median(res, lfq$factor_keys()[1], "sd")
```

plot_stat_violin	<i>plot Violin plot of sd CV or mean</i>
------------------	--

Description

plot Violin plot of sd CV or mean

Usage

```
plot_stat_violin(
  pdata,
  factor_keys_depth,
  stat = c("CV", "meanAbundance", "sd")
)
```

Arguments

pdata	data.frame with statistics
factor_keys_depth	character vector — factor columns for grouping
stat	either CV, mean or sd

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb1$data, bb1$config)
res <- lfq$get_Stats()$stats()
plot_stat_violin(res, lfq$relevant_factor_keys(), stat = "meanAbundance")
plot_stat_violin(res, lfq$relevant_factor_keys(), stat = "sd")
plot_stat_violin(res, lfq$relevant_factor_keys(), stat = "CV")
```

```
plot_stat_violin_median
```

plot Violin plot of sd CV or mean given intensity lower or above median

Description

plot Violin plot of sd CV or mean given intensity lower or above median

Usage

```
plot_stat_violin_median(
  pdata,
  factor_key,
  stat = c("CV", "meanAbundance", "sd")
)
```

Arguments

pdata	data.frame with statistics
factor_key	character — factor column name for x-axis
stat	either CV, mean or sd

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stdv_vs_mean\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb1$data, bb1$config)
res <- lfq$get_Stats()$stats()
plot_stat_violin_median(res, lfq$factor_keys()[1], stat = "meanAbundance")
```

plot_stdv_vs_mean	<i>plot stddev vs mean to asses stability of variance</i>
-------------------	---

Description

plot stddev vs mean to asses stability of variance

Usage

```
plot_stdv_vs_mean(pdata, factor_keys_depth, size = 2000)
```

Arguments

pdata	data.frame with statistics
factor_keys_depth	character vector — factor columns for faceting
size	how many points to sample (since scatter plot to slow for all)

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [summarize_stats\(\)](#)

Examples

```
bb1 <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb1$data, bb1$config)
res <- lfq$get_Stats()$stats()
plot_stdv_vs_mean(res, lfq$relevant_factor_keys())
```

prolfqua_data	<i>load data from prolfqua</i>
---------------	--------------------------------

Description

load data from prolfqua

Usage

```
prolfqua_data(datastr, package = "prolfqua")
```

Arguments

datastr	name of dataset
package	default prolfqua

Value

The computed result.

See Also

Other data: [PACKAGE_DATA](#), [data_checksummarizationrobust87](#), [data_checksummarizerobust69](#), [data_ionstar](#), [x5463yzwer453bbb](#)

Examples

```
ionstar <- prolfqua_data("data_ionstar")  
names(ionstar)
```

R6_extract_values	<i>Extract all value slots in an R6 object</i>
-------------------	--

Description

Extract all value slots in an R6 object

Usage

```
R6_extract_values(r6class)
```

Arguments

r6class	r6 class
---------	----------

Value

The computed result.

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
bb <- sim_lfq_data_peptide_config()
values <- R6_extract_values(bb$config)
names(values)
```

rank_peptide_by_intensity

ranks precursor - peptide by intensity.

Description

ranks precursor - peptide by intensity.

Usage

```
rank_peptide_by_intensity(pdata, response, hierarchy_keys)
```

Arguments

pdata data.frame
 response character — intensity column name
 hierarchy_keys character vector — all hierarchy column names

Value

data.frame

Examples

```
bb <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb$data, bb$config)
res <- rank_peptide_by_intensity(lfq$data_long(), lfq$response(), lfq$hierarchy_keys())
X <- res |> dplyr::select(c(lfq$hierarchy_keys(),
  srm_meanInt, srm_meanIntRank)) |> dplyr::distinct()
X |> dplyr::arrange(!!!rlang::syms(c(lfq$hierarchy_keys()[1], "srm_meanIntRank")))
```

register_facade	<i>Register a contrast facade class</i>
-----------------	---

Description

Adds an entry to the prolfqua facade registry so that [lookup_facade](#) (and downstream consumers such as `prolfquapp::R6_DEAnalyse$build_facade()`) can resolve it by short name. Intended for downstream packages that ship additional modelling backends (e.g. `prolfquasaint`) and want to plug into the same dispatch path as the built-in facades.

Usage

```
register_facade(
  name,
  class,
  needs,
  package = "prolfqua",
  needs_saint_annotation = FALSE,
  ...
)
```

Arguments

name	short method name, e.g. "saint".
class	character; R6 facade class name to instantiate.
needs	one of "same" (facade emits contrasts at the same hierarchy level as its input <i>e.g.</i> protein -> protein FC, peptide -> peptide FC) or "nested" (facade takes child-level input and emits parent-level contrasts, <i>e.g.</i> peptide -> protein FC).
package	package the facade class lives in. Defaults to "prolfqua".
needs_saint_annotation	TRUE if the backend requires annotation reading in SAINT mode (bait / control columns). Default FALSE.
...	additional fields stored on the registry entry, available to consumers via lookup_facade .

Value

The registered entry (invisibly).

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#),

```

build_model_limma_impute(), build_model_limma_voom(), build_model_limma_voom_impute(),
build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance_
compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(),
get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(),
impute_refit_singular(), linfoct_from_model(), linfoct_matrix_contrasts(), list_facades(),
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(),
sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(),
sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(),
summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

```

Examples

```

lookup_facade("lm")$class
# downstream packages call this from .onLoad():
# prolfqua::register_facade("saint", class = "ContrastsSAINTFacade",
#                               needs = "same", package = "prolfquasaint",
#                               needs_saint_annotation = TRUE)

```

response_matrix_as_tibble

Takes matrix of responses and converts into tibble

Description

Takes matrix of responses and converts into tibble

Usage

```
response_matrix_as_tibble(pdata, value, config, data = NULL, sep = "~lfq~")
```

Arguments

pdata	matrix with rownames encoding hierarchy keys
value	name of column to store values in
config	AnalysisConfiguration (needed for column name mapping)
data	optional data.frame to join back to
sep	separator used to unite the hierarchy keys

Examples

```

dd <- prolfqua::sim_lfq_data_peptide_config()
lfqdata <- prolfqua::LFQData$new(dd$data, dd$config)
res <- tidy_to_wide_config(lfqdata, as.matrix = TRUE)

res_scaled <- scale(res$data)
xx <- response_matrix_as_tibble(
  res_scaled, "srm_intensityScaled", lfqdata$get_config()
)
xx <- response_matrix_as_tibble(
  res_scaled, "srm_intensityScaled",
  lfqdata$get_config(), lfqdata$data_long()
)

```

)

robust_scale	<i>robust scale wrapper</i>
--------------	-----------------------------

Description

robust scale wrapper

Usage

robust_scale(data, dim = 2, preserve_mean = FALSE)

See Also

Other preprocessing: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [apply_to_response_matrix\(\)](#), [filter_proteins_by_peptide\(\)](#), [get_robscales\(\)](#), [nr_B_in_A\(\)](#), [scale_with_subset\(\)](#)

Examples

```
mat <- matrix(c(1, 2, 3, 4, 10, 12), nrow = 3)
scaled <- robust_scale(mat)
dim(scaled)
```

sample_subset	<i>Sample subset of proteins/peptides/precursors</i>
---------------	--

Description

Sample subset of proteins/peptides/precursors

Usage

sample_subset(size, pdata, hierarchy_keys_depth)

Arguments

size	size of sample
pdata	tidy table
hierarchy_keys_depth	character vector — hierarchy columns at current depth

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_config\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
bb <- sim_lfq_data_peptide_config(Nprot = 5)
subset <- sample_subset(2, bb$data, bb$config$hierarchy_keys_depth())
length(unique(subset[[bb$config$hierarchy_keys_depth()[1]]]))
```

scale_with_subset	<i>Scale data using a subset of the data</i>
-------------------	--

Description

this should reduce the overall variance.

Usage

```
scale_with_subset(lfqdata, lfqsubset, preserve_mean = FALSE, colname = NULL)
```

Arguments

lfqdata	LFQData object with full dataset
lfqsubset	LFQData object with subset for computing scales
preserve_mean	default FALSE - sets mean to zero
colname	name of the scaled intensity column; default NULL uses <response>_subset_scaled

Value

list with data, scales, and colname

See Also

Other preprocessing: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [apply_to_response_matrix\(\)](#), [filter_proteins_by_peptide](#), [get_rob scales\(\)](#), [nr_B_in_A\(\)](#), [robust_scale\(\)](#)

Examples

```
bb <- sim_lfq_data_peptide_config(Nprot = 100)
lfqdata <- LFQData$new(bb$data, bb$config)
lfqdata <- lfqdata$get_Transformer()$log2()$lfq
s1 <- get_rob scales(lfqdata)
res <- scale_with_subset(lfqdata, lfqdata)
cfg <- lfqdata$get_config()$clone(deep = TRUE)
cfg$set_response(res$colname)
lfqres <- LFQData$new(res$data, cfg)
s2 <- get_rob scales(lfqres)
stopifnot(abs(mean(s1$mads) - mean(s2$mads)) < 1e-6)
```

script_copy_helper_vec

copy script files and other from a package to workdir

Description

copy script files and other from a package to workdir

Usage

```
script_copy_helper_vec(
  runscripts,
  workdir = getwd(),
  packagename = "prolfqua",
  overwrite = TRUE
)
```

Arguments

runscripts	paths of the files relative to the package root (or inst/)
workdir	directory to copy the files to
packagename	package to copy the files from
overwrite	overwrite existing files in workdir, default TRUE

Value

paths of the copied files

Examples

```
copied <- script_copy_helper_vec("CITATION", workdir = tempdir())
basename(copied)
```

separate_hierarchy

Separates hierarchy columns into starting columns

Description

Separates hierarchy columns into starting columns

Usage

```
separate_hierarchy(data, config)
```

Arguments

data	data.frame
config	AnalysisConfiguration

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_config\(\)](#), [sample_subset\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
bb <- sim_lfq_data_protein_config()
dt <- separate_hierarchy(bb$data, bb$config)
base::setdiff(colnames(dt), colnames(bb$data))
stopifnot(ncol(dt) >= ncol(bb$data))
```

setup_analysis

Setup a tidy table compatible with a [AnalysisConfiguration](#)

Description

Extracts columns relevant for a configuration from a data frame and create new columns e.g. sampleName column etc.

Usage

```
setup_analysis(data, configuration, debug = FALSE)
```

Arguments

data	data.frame
configuration	AnalysisConfiguration
debug	if FALSE (default) and any hierarchy-key/sample combination has more than one observation, stop with an informative error. If TRUE, warn and return the diagnostic count table (inspect rows where n > 1) instead.

Value

A tibble with the columns relevant for configuration (sample, hierarchy, factor and response columns). Stops if any hierarchy-key/sample combination is observed more than once (unless debug = TRUE).

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_config\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [table_factors\(\)](#), [table_factors_size\(\)](#)

Examples

```
# Example with a normValue column (e.g., creatinine)
set.seed(1234)
data <- sim_lfq_data(Nprot = 10, PEPTIDE = TRUE, N = 4)
data$nr_children <- 1
data$isotopeLabel <- "light"
data$qValue <- 0

# Add creatinine values per sample (not per protein/peptide)
sample_creatinine <- data |> dplyr::select(sample) |> dplyr::distinct() |>
  dplyr::mutate(creatinine = rnorm(dplyr::n(), mean = 100, sd = 10))
data <- dplyr::inner_join(data, sample_creatinine, by = "sample")

config <- AnalysisConfiguration$new()
config$file_name = "sample"
config$factors["group_"] = "group"
config$hierarchy[["protein_Id"]] = c("proteinID", "idtype2")
config$hierarchy[["peptide_Id"]] = "peptideID"
config$set_response("abundance")
config$norm_value = "creatinine"

adata <- setup_analysis(data, config)
```

sigma.rfit_prolfqua	<i>Scale estimate for an augmented rfit fit</i>
---------------------	---

Description

Returns the rank-based scale ‘tauhat’. This stands in for the residual standard deviation of an ‘lm’; it does not affect the Wald std.error or p-value (those come from ‘vcov()’), but it is reported and used as the per- protein variance estimate if results are passed to limma/eBayes moderation.

Usage

```
## S3 method for class 'rfit_prolfqua'
sigma(object, ...)
```

Arguments

object	an ‘rfit_prolfqua’ fit
...	ignored

Value

the rank-based scale estimate ‘tauhat’.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNest](#)

ContrastsLmerNestedFacade, ContrastsModerated, ContrastsModeratedDEqMS, ContrastsPlotter, ContrastsRLMFacade, ContrastsROPECA, ContrastsROPECANestedFacade, ContrastsRfitFacade, ContrastsRfitImputeFacade, ContrastsTable, INTERNAL_FUNCTIONS_BY_FAMILY, LR_test(), Model, ModelFirth, ModelLimma, StrategyLM, StrategyLimma, StrategyLimpa, StrategyLmer, StrategyLogistf, StrategyRLM, StrategyRfit, build_contrast_analysis(), build_model(), build_model_glm_peptide(), build_model_glm_protein(), build_model_impute(), build_model_limma(), build_model_limma_impute(), build_model_limma_voom(), build_model_limma_voom_impute(), build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance_compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

Examples

```
fit <- strategy_rfit("Sepal.Length ~ Species")$model_fun(iris)
sigma(fit)
```

sigma.rlm	<i>Residual scale estimate for rlm objects</i>
-----------	--

Description

stats::sigma returns NA for **rlm** objects. This S3 method computes a weighted residual scale estimate instead.

Usage

```
## S3 method for class 'rlm'
sigma(object, ...)
```

Arguments

object	an rlm object
...	ignored

Value

numeric scalar

sim_build_models_lm	<i>build dataframe with models for testing</i>
---------------------	--

Description

build dataframe with models for testing

Usage

```
sim_build_models_lm(
  model = c("parallel2", "parallel3", "factors", "interaction"),
  Nprot = 10,
  with_missing = TRUE,
  weight_missing = 1
)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
modi <- sim_build_models_lm(model = "interaction", weight_missing = 1)
stopifnot(dim(modi$model_df) == c(10,9))
mod2 <- sim_build_models_lm(model = "parallel2", weight_missing = 1)
mod2$model_df$linear_model[[1]]
mod3 <- sim_build_models_lm(model = "parallel3", weight_missing = 1)
modf <- sim_build_models_lm(model = "factors", weight_missing = 1)
```

sim_build_models_lmer *build lmer model from simulated data*

Description

build lmer model from simulated data

Usage

```
sim_build_models_lmer(
  model = c("parallel2", "parallel3", "factors", "interaction"),
  Nprot = 10,
  with_missing = TRUE,
  weight_missing = 1
)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
mod1 <- sim_build_models_lmer(model = "interaction", weight_missing = 1)
stopifnot(sum(mod1$model_df$has_model_fit) == 6)
mod2 <- sim_build_models_lmer(model = "parallel2", weight_missing = 1)
stopifnot(sum(mod2$model_df$has_model_fit) == 6)
mod4 <- sim_build_models_lmer(model = "parallel3", weight_missing = 1)
stopifnot(sum(mod4$model_df$has_model_fit) == 6)
modf <- sim_build_models_lmer(model = "factors", weight_missing = 1)
stopifnot(sum(modf$model_df$has_model_fit) == 6)
```

sim_build_models_logistf

build dataframe with models for testing

Description

build dataframe with models for testing

Usage

```
sim_build_models_logistf(
  model = c("parallel2", "parallel3", "factors", "interaction"),
  Nprot = 10,
  with_missing = TRUE,
  weight_missing = 1,
  peptide = FALSE
)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
mod1 <- sim_build_models_logistf(model = "interaction", weight_missing = 1)
stopifnot(dim(mod1$model_df) == c(10,9))
mod2 <- sim_build_models_logistf(model = "parallel2", weight_missing = 1)
mod2$model_df$linear_model[[1]]
mod3 <- sim_build_models_logistf(model = "parallel3", weight_missing = 1)
modf <- sim_build_models_logistf(model = "factors", weight_missing = 1)

mod3 <- sim_build_models_logistf(model = "parallel3", weight_missing = 1, peptide=TRUE)
modf <- sim_build_models_logistf(model = "factors", weight_missing = 1, peptide=TRUE)
```

sim_lfq_data

*simulate protein level data with two groups***Description**

simulate protein level data with two groups

Usage

```
sim_lfq_data(
  Nprot = 20,
  N = 4,
  fc = list(A = c(D = -2, U = 2, N = 0), B = c(D = 1, U = -4)),
  prop = list(A = c(D = 10, U = 10), B = c(D = 5, U = 20)),
  mean_prot = 20,
  sdlog = log(1.2),
  probability_of_success = 0.3,
  PEPTIDE = FALSE
)
```

Arguments

Nprot	number of proteins
N	group size
fc	D - down regulation N - matrix, U - regulation
prop	proportion of down (D), up (U) and not regulated (N)
mean_prot	mean protein abundance on the natural scale
sdlog	standard deviation of the log-normal abundance distribution
probability_of_success	probability of success for the geometric distribution of peptide counts
PEPTIDE	if TRUE, simulate peptide-level data; if FALSE, protein-level

Value

A data frame containing simulated LFQ data.

Examples

```
res <- sim_lfq_data(Nprot = 10)
res <- sim_lfq_data(Nprot = 10, PEPTIDE = TRUE)
res <- sim_lfq_data(Nprot = 10, N=4)
```

```
sim_lfq_data_2factor_config
```

Simulate data, protein, with config with 2 factors Treatment and Background

Description

Simulate data, protein, with config with 2 factors Treatment and Background

Usage

```
sim_lfq_data_2factor_config(
  Nprot = 10,
  with_missing = TRUE,
  weight_missing = 0.2,
  PEPTIDE = FALSE,
  seed = 1234,
  TWO = TRUE
)
```

Arguments

Nprot	number of proteins to simulate
with_missing	add missing values, default TRUE
weight_missing	controls proportion of missing values; greater weight means more missingness
PEPTIDE	if TRUE, simulate peptide-level data; if FALSE, protein-level
seed	seed for reproducibility, if NULL no seed is set.
TWO	use two factors for modelling

Value

A list containing simulated data and an analysis configuration.

Examples

```
x <- sim_lfq_data_2factor_config(PEPTIDE= FALSE)
dim(x$data)
stopifnot("data.frame" %in% class(x$data))
stopifnot("AnalysisConfiguration" %in% class(x$config))
x <- sim_lfq_data_2factor_config(PEPTIDE = TRUE)

head(x$data)
x <- sim_lfq_data_2factor_config(PEPTIDE = TRUE, TWO = TRUE)
names(x)
nrow(x$data) > 10
x$data$Treatment |> table()
```

```
sim_lfq_data_peptide_config
```

Simulate data, protein and peptide, with config

Description

Simulate data, protein and peptide, with config

Usage

```
sim_lfq_data_peptide_config(
  Nprot = 10,
  with_missing = TRUE,
  weight_missing = 0.2,
  seed = 1234,
  N = 4
)
```

Arguments

Nprot	number of proteins to simulate
with_missing	add missing values, default TRUE
weight_missing	controls proportion of missing values; greater weight means more missingness
seed	seed for reproducibility, if NULL no seed is set.
N	number of replicates per group

Value

A list containing simulated data and an analysis configuration.

Examples

```
x <- sim_lfq_data_peptide_config()
stopifnot("data.frame" %in% class(x$data))
stopifnot("AnalysisConfiguration" %in% class(x$config))
```

```
sim_lfq_data_protein_config
```

Simulate data, protein, with config

Description

Simulate data, protein, with config

Usage

```
sim_lfq_data_protein_config(
  Nprot = 10,
  with_missing = TRUE,
  weight_missing = 0.2,
  seed = 1234,
  paired = FALSE
)
```

Arguments

Nprot	number of proteins to simulate
with_missing	add missing values, default TRUE
weight_missing	controls proportion of missing values; greater weight means more missingness
seed	seed for reproducibility, if NULL no seed is set.
paired	if TRUE, add a paired subject factor to the design

Value

A list containing simulated data and an analysis configuration.

Examples

```
x <- sim_lfq_data_protein_config()
stopifnot("data.frame" %in% class(x$data))
stopifnot("AnalysisConfiguration" %in% class(x$config))
x <- sim_lfq_data_protein_config(with_missing = FALSE)

stopifnot(sum(is.na(x$data$abundance)) == 0)
xp <- sim_lfq_data_protein_config(with_missing = FALSE, paired = TRUE)
stopifnot(length(xp$config$factors) == 2)
stopifnot(nrow(xp$data) == nrow(x$data))
```

sim_make_model_lm	<i>make interaction model for examples</i>
-------------------	--

Description

make interaction model for examples

Usage

```
sim_make_model_lm(
  model = c("parallel2", "parallel3", "factors", "interaction")
)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
m <- sim_make_model_lm()
mi <- sim_make_model_lm("interaction")
stopifnot(length(coefficients(summary(mi))[, "Estimate"]) == 4)
mf <- sim_make_model_lmer("factors")
m2 <- sim_make_model_lmer("parallel2")
m3 <- sim_make_model_lmer("parallel3")
```

sim_make_model_lmer *make interaction model for examples*

Description

make interaction model for examples

Usage

```
sim_make_model_lmer(
  model = c("parallel2", "parallel3", "factors", "interaction")
)
```

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#),

[ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
mf <- sim_make_model_lmer("factors")
mi <- sim_make_model_lmer("interaction")
```

StrategyBase	<i>Base class for per-subject model strategies</i>
--------------	--

Description

Base class for per-subject model strategies
Base class for per-subject model strategies

Details

Holds the fields, constructor and default methods shared by [StrategyLM](#), [StrategyRLM](#), [StrategyLmer](#), [StrategyRfit](#), [StrategyLogistf](#) and [StrategyBinomial](#). A subclass implements the private `fit(x)` method and overrides only the methods that differ.

Value

An R6 class generator.

Public fields

`formula` model formula
`model_name` name of model
`anova_df` ANOVA extractor, see [AnovaExtractor](#)

Methods

Public methods:

- [StrategyBase\\$new\(\)](#)
- [StrategyBase\\$model_fun\(\)](#)
- [StrategyBase\\$isSingular\(\)](#)
- [StrategyBase\\$contrast_fun\(\)](#)
- [StrategyBase\\$df_residual\(\)](#)
- [StrategyBase\\$sigma\(\)](#)
- [StrategyBase\\$clone\(\)](#)

Method `new()`: Create a new strategy

Usage:

```
StrategyBase$new(modelstr, model_name = "Model")
```

Arguments:

`modelstr` model formula string

`model_name` name of model

Method `model_fun()`: Fit the model to one subject's data. A failed fit returns the error message.

Usage:

```
StrategyBase$model_fun(x, pb)
```

Arguments:

`x` data.frame for one subject

`pb` optional progress bar

Method `isSingular()`: Check if model is singular (NA coefficients or $df < 2$)

Usage:

```
StrategyBase$isSingular(model)
```

Arguments:

`model` fitted model

Method `contrast_fun()`: Compute contrasts from fitted model

Usage:

```
StrategyBase$contrast_fun(...)
```

Arguments:

`...` passed to [compute_contrast](#)

Method `df_residual()`: Get residual degrees of freedom

Usage:

```
StrategyBase$df_residual(model)
```

Arguments:

`model` fitted model

Method `sigma()`: Get residual standard error

Usage:

```
StrategyBase$sigma(model)
```

Arguments:

model fitted model

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
StrategyBase$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

StrategyBinomial

Quasibinomial detection-count strategy

Description

Quasibinomial detection-count strategy

Quasibinomial detection-count strategy

Details

Fits detected and undetected child-feature counts for each parent feature using [glm](#) with a quasibinomial family. The symmetric pseudo-count stabilizes fits under complete separation; it is not equivalent to Firth's bias-reducing penalty.

Value

An R6 class generator.

Super class

```
prolfqua::StrategyBase -> StrategyBinomial
```

Public fields

prior_count symmetric pseudo-count added to both outcomes

Methods

Public methods:

- [StrategyBinomial\\$new\(\)](#)
- [StrategyBinomial\\$clone\(\)](#)

Method new(): Create a quasibinomial count strategy. `sigma()` is the Pearson residual scale used by `vcov()`.

Usage:

```
StrategyBinomial$new(
  modelstr,
  prior_count = 0.1,
  model_name = "binomial_nested"
)
```

Arguments:

`modelstr` right-hand-side model formula, for example "`~ group_`"
`prior_count` non-negative symmetric pseudo-count
`model_name` model identity

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
StrategyBinomial$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
dat <- data.frame(
  group_ = factor(rep(c("A", "B"), each = 4)),
  detected = c(1, 2, 1, 3, 4, 5, 3, 5),
  undetected = c(4, 3, 4, 2, 1, 0, 2, 0)
)
strategy <- StrategyBinomial$new("~ group_")
fit <- strategy$model_fun(dat)
coefficients(fit)
```

StrategyLimma

R6 class for limma modelling strategy

Description

R6 class for limma modelling strategy

R6 class for limma modelling strategy

Details

Analogous to [strategy_lm](#) but for limma's matrix-based pipeline. Consumed by [build_model_limma](#).

Value

An R6 class generator.

Public fields

`formula` model formula

`model_name` name of model

`trend` logical, passed to [eBayes](#)

`robust` logical, passed to [eBayes](#)

`weights` either a character string (column name) or a numeric matrix

Methods

Public methods:

- [StrategyLimma\\$new\(\)](#)
- [StrategyLimma\\$clone\(\)](#)

Method `new()`: Create a new StrategyLimma

Usage:

```
StrategyLimma$new(
  modelstr,
  model_name = "limma",
  trend = FALSE,
  robust = FALSE,
  weights = NULL
)
```

Arguments:

`modelstr`, `model_name`, `trend`, `robust`, `weights` see [strategy_limma](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
StrategyLimma$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- StrategyLimma$new("abundance ~ group")
strat$formula
strat$model_name
```

StrategyLimpa

*R6 class for limpa modelling strategy***Description**

R6 class for limpa modelling strategy

R6 class for limpa modelling strategy

Details

Consumed by [build_model_limpa](#). Configures the `limpa::voomaLmFitWithImputation` call: formula for the design matrix, trend/robust flags for [eBayes](#), and optional vooma plotting parameters.

Value

An R6 class generator.

Public fields

formula model formula

model_name name of model

trend logical, passed to [eBayes](#)robust logical, passed to [eBayes](#)

plot logical, plot the vooma mean-variance trend

span lowess smoother span for vooma trend (NULL = auto)

Methods**Public methods:**

- [StrategyLimpa\\$new\(\)](#)
- [StrategyLimpa\\$clone\(\)](#)

Method `new()`: Create a new StrategyLimpa*Usage:*

```
StrategyLimpa$new(
  modelstr,
  model_name = "limpa",
  trend = FALSE,
  robust = FALSE,
  plot = FALSE,
  span = NULL
)
```

*Arguments:*modelstr, model_name, trend, robust, plot, span see [strategy_limpa](#)**Method** `clone()`: The objects of this class are cloneable with this method.*Usage:*

```
StrategyLimpa$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- StrategyLimpafacade$new("abundance ~ group_")
strat$formula
strat$model_name
```

StrategyLM	Linear model strategy (R6 class)
------------	----------------------------------

Description

Linear model strategy (R6 class)

Linear model strategy (R6 class)

Details

Encapsulates everything needed to fit per-protein linear models and extract contrasts: the formula, model fitting function, singularity check, contrast computation, ANOVA, and residual statistics.

Value

An R6 class generator.

Super class

```
prolfqua::StrategyBase -> StrategyLM
```

Public fields

`weights` optional character string naming a column in the data containing per-observation weights, passed to `lm`.

Methods**Public methods:**

- [StrategyLM\\$new\(\)](#)
- [StrategyLM\\$clone\(\)](#)

Method `new()`: Create a new StrategyLM

Usage:

```
StrategyLM$new(modelstr, model_name = "Model", weights = NULL)
```

Arguments:

`modelstr` model formula string

`model_name` name of model

`weights` optional character string naming a column in the data containing per-observation weights

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
StrategyLM$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- StrategyLM$new("Intensity ~ condition", model_name = "parallel design")
strat$formula
strat$weights
```

StrategyLmer

*Linear mixed-effects model strategy (R6 class)***Description**

Linear mixed-effects model strategy (R6 class)

Linear mixed-effects model strategy (R6 class)

Details

Encapsulates everything needed to fit per-protein linear mixed-effects models via [lmer](#) and extract contrasts.

Value

An R6 class generator.

Super class

[prolfqua::StrategyBase](#) -> StrategyLmer

Methods**Public methods:**

- [StrategyLmer\\$isSingular\(\)](#)
- [StrategyLmer\\$contrast_fun\(\)](#)
- [StrategyLmer\\$clone\(\)](#)

Method [isSingular\(\)](#): Check if model is singular

Usage:

[StrategyLmer\\$isSingular\(model\)](#)

Arguments:

`model` fitted model

Method [contrast_fun\(\)](#): Compute contrasts from fitted model

Usage:

[StrategyLmer\\$contrast_fun\(...\)](#)

Arguments:

`...` passed to [compute_lmer_contrast](#)

Method [clone\(\)](#): The objects of this class are cloneable with this method.

Usage:

[StrategyLmer\\$clone\(deep = FALSE\)](#)

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImpu](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNeste](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
istar <- prolfqua::sim_lfq_data_peptide_config(Nprot = 10, with_missing = FALSE)
istar <- prolfqua::LFQData$new(istar$data, istar$config)
istar$set_data(istar$data_long() |> dplyr::group_by(protein_Id) |>
  dplyr::mutate(abundanceC = abundance - mean(abundance)) |> dplyr::ungroup())
strat <- StrategyLmer$new("abundanceC ~ group_ + (1|peptide_Id)",
  model_name = "random_example")
mod <- build_model(istar, strat)
sum(mod$model_df$has_model_fit)
```

StrategyLogistf	<i>Firth's logistic regression strategy (R6 class)</i>
-----------------	--

Description

Firth's logistic regression strategy (R6 class)
Firth's logistic regression strategy (R6 class)

Details

Encapsulates everything needed to fit per-protein Firth's bias-reduced logistic regression via [logistf](#) and extract contrasts.

Value

An R6 class generator.

Super class

```
prolfqua::StrategyBase -> StrategyLogistf
```

Methods**Public methods:**

- [StrategyLogistf\\$new\(\)](#)
- [StrategyLogistf\\$df_residual\(\)](#)
- [StrategyLogistf\\$sigma\(\)](#)
- [StrategyLogistf\\$clone\(\)](#)

Method `new()`: Create a new StrategyLogistf

Usage:

```
StrategyLogistf$new(modelstr, model_name = "logistf")
```

Arguments:

`modelstr` model formula string

`model_name` name of model

Method `df_residual()`: Get residual degrees of freedom

Usage:

```
StrategyLogistf$df_residual(model)
```

Arguments:

`model` fitted model

Method `sigma()`: Get residual standard error (always 1 for logistic)

Usage:

```
StrategyLogistf$sigma(model)
```

Arguments:

`model` fitted model

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
StrategyLogistf$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#),

```
build_model_limma_voom(), build_model_limma_voom_impute(), build_model_limpa(), build_model_logistf,
compute_borrowed_variance(), compute_borrowed_variance_limma(), compute_contrast(),
compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(),
get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(),
linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_res,
model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(),
new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(),
sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(),
sim_make_model_lmer(), strategy_limma(), strategy_limpa(), strategy_logistf(), summary_ROPECA_median,
vcov.rfit_prolfqua()
```

Examples

```
strat <- StrategyLogistf$new("bin_resp ~ condition")
strat$formula
```

StrategyRfit	Rank-based regression strategy (R6 class)
--------------	---

Description

Rank-based regression strategy (R6 class)
Rank-based regression strategy (R6 class)

Details

Encapsulates everything needed to fit per-protein rank-based linear models via [rfit](#) and extract Wald contrasts. The fit is augmented (model frame, terms, `rfit_prolfqua` subclass) so it satisfies the same `coef()` / `vcov()` / `terms()` contract as `lm`.
Unlike [StrategyLM](#), `rfit` takes no observation weights, so `nr_children` weighting is not supported by this backend.

Value

An R6 class generator.

Super class

```
prolfqua::StrategyBase -> StrategyRfit
```

Methods

Public methods:

- [StrategyRfit\\$new\(\)](#)
- [StrategyRfit\\$clone\(\)](#)

Method `new()`: Create a new `StrategyRfit`. Residual df and scale come from `df.residual()` and `sigma()` of the augmented fit.

Usage:

```
StrategyRfit$new(modelstr, model_name = "rfit")
```

Arguments:
modelstr model formula string
model_name name of model

Method clone(): The objects of this class are cloneable with this method.

Usage:
StrategyRfit\$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- StrategyRfit$new("Intensity ~ condition", model_name = "parallel design")
strat$formula
```

StrategyRLM	<i>Robust linear model strategy (R6 class)</i>
-------------	--

Description

Robust linear model strategy (R6 class)
Robust linear model strategy (R6 class)

Details

Encapsulates everything needed to fit per-protein robust linear models via [rlm](#) and extract contrasts.

Value

An R6 class generator.

Super class

`prolfqua::StrategyBase` -> StrategyRLM

Methods**Public methods:**

- `StrategyRLM$sigma()`
- `StrategyRLM$clone()`

Method `sigma()`: Get the robust scale estimate used by `vcov()`

Returns `model$s`, the robust scale MASS::`rlm` builds its `vcov()` / `summary()` standard errors from. This must match the scale embedded in `vcov()` so that `ContrastsModerated`'s `sigma` / `sqrt(var.post)` rescaling stays coherent; `stats::sigma()` returns the ordinary-residual scale instead and would distort moderated `rlm` statistics.

Usage:

`StrategyRLM$sigma(model)`

Arguments:

`model` fitted model

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`StrategyRLM$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimp](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limp\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_limma\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_res\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limp\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- StrategyRLM$new("Intensity ~ condition", model_name = "parallel design")
strat$formula
```

strategy_binomial	Create a quasibinomial detection-count strategy
-------------------	---

Description

Create a quasibinomial detection-count strategy

Usage

```
strategy_binomial(modelstr, prior_count = 0.1, model_name = "binomial_nested")
```

Arguments

modelstr	right-hand-side model formula
prior_count	non-negative symmetric pseudo-count
model_name	model identity

Value

A [StrategyBinomial](#) object.

Examples

```
strategy <- strategy_binomial("~ group_", prior_count = 0.1)
strategy$formula
```

strategy_limma	Create limma modelling strategy
----------------	---------------------------------

Description

Wrapper that returns a [StrategyLimma](#) R6 object.

Usage

```
strategy_limma(
  modelstr,
  model_name = "limma",
  trend = FALSE,
  robust = FALSE,
  weights = NULL
)
```

Arguments

modelstr	model formula as string (e.g. "abundance ~ group_")
model_name	name of model
trend	logical, passed to eBayes
robust	logical, passed to eBayes
weights	either a character string (column name in annotation for per-sample weights) or a numeric matrix (proteins x samples) passed to lmFit . Default NULL (no weights).

Value

The computed result.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpa](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpa\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limpa\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- strategy_limpa("abundance ~ group_")
strat$formula
strat$model_name
```

strategy_limpa

Create limpa modelling strategy

Description

Wrapper that returns a [StrategyLimpa](#) R6 object.

Usage

```
strategy_limpa(
  modelstr,
  model_name = "limpa",
  trend = FALSE,
  robust = FALSE,
  plot = FALSE,
  span = NULL
)
```

Arguments

modelstr	model formula as string (e.g. "abundance ~ group_")
model_name	name of model
trend	logical, passed to eBayes
robust	logical, passed to eBayes
plot	logical, plot the vooma mean-variance trend
span	lowess smoother span (NULL = auto)

Value

The computed result.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_2\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
strat <- strategy_limpa("abundance ~ group_")
strat$formula
strat$model_name
```

strategy_logistf	Create Firth's logistic regression strategy
------------------	---

Description

Convenience wrapper that creates a [StrategyLogistf](#) object.

Convenience wrapper that creates a [StrategyRfit](#) object.

Convenience wrapper that creates a [StrategyLmer](#) object.

Convenience wrapper that creates a [StrategyLM](#) object.

Convenience wrapper that creates a [StrategyRLM](#) object.

Usage

```
strategy_logistf(modelstr, model_name = "logistf")
```

```
strategy_rfit(modelstr, model_name = "rfit")
```

```
strategy_lmer(modelstr, model_name = "Model")
```

```
strategy_lm(modelstr, model_name = "Model", weights = NULL)
```

```
strategy_rlm(modelstr, model_name = "Model")
```

Arguments

modelstr	model formula
model_name	name of model
weights	optional character string naming a column in the data containing per-observation weights, passed to lm . Default NULL (unweighted).

Value

a [StrategyLogistf](#) object

a [StrategyRfit](#) object

a [StrategyLmer](#) object

a [StrategyLM](#) object

a [StrategyRLM](#) object

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#),

```

StrategyLogistf, StrategyRLM, StrategyRfit, build_contrast_analysis(), build_model(),
build_model_glm_peptide(), build_model_glm_protein(), build_model_impute(), build_model_limma(),
build_model_limma_impute(), build_model_limma_voom(), build_model_limma_voom_impute(),
build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance(),
compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(),
get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(),
impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(),
lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(),
moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(),
register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(),
sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(),
strategy_limpa(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

```

Other modelling: AnovaExtractor, Contrasts, ContrastsDEqMSFacade, ContrastsDEqMSVoomFacade, ContrastsFacadeBase, ContrastsFirth, ContrastsFirthFacade, ContrastsFirthNestedFacade, ContrastsLMFacade, ContrastsLMImputeFacade, ContrastsLimma, ContrastsLimmaFacade, ContrastsLimmaImputeFacade, ContrastsLimmaVoomFacade, ContrastsLimmaVoomImputeFacade, ContrastsLimpaFacade, ContrastsLimpaNestedFacade, ContrastsLmerNestedFacade, ContrastsModerated, ContrastsModeratedDEqMS, ContrastsPlotter, ContrastsRLMFacade, ContrastsROPECA, ContrastsROPECANestedFacade, ContrastsRfitFacade, ContrastsRfitImputeFacade, ContrastsTable, INTERNAL_FUNCTIONS_BY_FAMILY, LR_test(), Model, ModelFirth, ModelLimma, StrategyLM, StrategyLimma, StrategyLimpa, StrategyLmer, StrategyLogistf, StrategyRLM, StrategyRfit, build_contrast_analysis(), build_model(), build_model_glm_peptide(), build_model_glm_protein(), build_model_impute(), build_model_limma(), build_model_limma_impute(), build_model_limma_voom(), build_model_limma_voom_impute(), build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance(), compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

Other modelling: AnovaExtractor, Contrasts, ContrastsDEqMSFacade, ContrastsDEqMSVoomFacade, ContrastsFacadeBase, ContrastsFirth, ContrastsFirthFacade, ContrastsFirthNestedFacade, ContrastsLMFacade, ContrastsLMImputeFacade, ContrastsLimma, ContrastsLimmaFacade, ContrastsLimmaImputeFacade, ContrastsLimmaVoomFacade, ContrastsLimmaVoomImputeFacade, ContrastsLimpaFacade, ContrastsLimpaNestedFacade, ContrastsLmerNestedFacade, ContrastsModerated, ContrastsModeratedDEqMS, ContrastsPlotter, ContrastsRLMFacade, ContrastsROPECA, ContrastsROPECANestedFacade, ContrastsRfitFacade, ContrastsRfitImputeFacade, ContrastsTable, INTERNAL_FUNCTIONS_BY_FAMILY, LR_test(), Model, ModelFirth, ModelLimma, StrategyLM, StrategyLimma, StrategyLimpa, StrategyLmer, StrategyLogistf, StrategyRLM, StrategyRfit, build_contrast_analysis(), build_model(), build_model_glm_peptide(), build_model_glm_protein(), build_model_impute(), build_model_limma(), build_model_limma_impute(), build_model_limma_voom(), build_model_limma_voom_impute(), build_model_limpa(), build_model_logistf(), compute_borrowed_variance(), compute_borrowed_variance(), compute_contrast(), compute_lmer_contrast(), df.residual.rfit_prolfqua(), get_anova_df(), get_complete_model_fit(), get_p_values_pbeta(), group_label(), impute_from_model(), impute_refit_singular(), linfct_from_model(), linfct_matrix_contrasts(), list_facades(), lookup_facade(), merge_contrasts_results(), model_analyse(), model_summary(), moderated_p_deqms(), moderated_p_deqms_long(), moderated_p_limma(), new_imputed_model(), pivot_model_contrasts_to_wide(), register_facade(), sigma.rfit_prolfqua(), sim_build_models_lm(), sim_build_models_lmer(), sim_build_models_logistf(), sim_make_model_lm(), sim_make_model_lmer(), strategy_limma(), strategy_limpa(), summary_ROPECA_median_p_scaled(), vcov.rfit_prolfqua()

Other modelling: `AnovaExtractor`, `Contrasts`, `ContrastsDEqMSFacade`, `ContrastsDEqMSVoomFacade`, `ContrastsFacadeBase`, `ContrastsFirth`, `ContrastsFirthFacade`, `ContrastsFirthNestedFacade`, `ContrastsLMFacade`, `ContrastsLMImputeFacade`, `ContrastsLimma`, `ContrastsLimmaFacade`, `ContrastsLimmaImputeFacade`, `ContrastsLimmaVoomFacade`, `ContrastsLimmaVoomImputeFacade`, `ContrastsLimpafacade`, `ContrastsLimpaNestedFacade`, `ContrastsLmerNestedFacade`, `ContrastsModerated`, `ContrastsModeratedDEqMS`, `ContrastsPlotter`, `ContrastsRLMFacade`, `ContrastsROPECA`, `ContrastsROPECANestedFacade`, `ContrastsRfitFacade`, `ContrastsRfitImputeFacade`, `ContrastsTable`, `INTERNAL_FUNCTIONS_BY_FAMILY`, `LR_test()`, `Model`, `ModelFirth`, `ModelLimma`, `StrategyLM`, `StrategyLimma`, `StrategyLimpafacade`, `StrategyLmer`, `StrategyLogistf`, `StrategyRLM`, `StrategyRfit`, `build_contrast_analysis()`, `build_model()`, `build_model_glm_peptide()`, `build_model_glm_protein()`, `build_model_impute()`, `build_model_limma()`, `build_model_limma_impute()`, `build_model_limma_voom()`, `build_model_limma_voom_impute()`, `build_model_limpafacade()`, `build_model_logistf()`, `compute_borrowed_variance()`, `compute_borrowed_variance_contrast()`, `compute_lmer_contrast()`, `df.residual.rfit_prolfqua()`, `get_anova_df()`, `get_complete_model_fit()`, `get_p_values_pbeta()`, `group_label()`, `impute_from_model()`, `impute_refit_singular()`, `linfct_from_model()`, `linfct_matrix_contrasts()`, `list_facades()`, `lookup_facade()`, `merge_contrasts_results()`, `model_analyse()`, `model_summary()`, `moderated_p_deqms()`, `moderated_p_deqms_long()`, `moderated_p_limma()`, `new_imputed_model()`, `pivot_model_contrasts_to_wide()`, `register_facade()`, `sigma.rfit_prolfqua()`, `sim_build_models_lm()`, `sim_build_models_lmer()`, `sim_build_models_logistf()`, `sim_make_model_lm()`, `sim_make_model_lmer()`, `strategy_limma()`, `strategy_limpafacade()`, `summary_ROPECA_median_p_scaled()`, `vcov.rfit_prolfqua()`

Other modelling: `AnovaExtractor`, `Contrasts`, `ContrastsDEqMSFacade`, `ContrastsDEqMSVoomFacade`, `ContrastsFacadeBase`, `ContrastsFirth`, `ContrastsFirthFacade`, `ContrastsFirthNestedFacade`, `ContrastsLMFacade`, `ContrastsLMImputeFacade`, `ContrastsLimma`, `ContrastsLimmaFacade`, `ContrastsLimmaImputeFacade`, `ContrastsLimmaVoomFacade`, `ContrastsLimmaVoomImputeFacade`, `ContrastsLimpafacade`, `ContrastsLimpaNestedFacade`, `ContrastsLmerNestedFacade`, `ContrastsModerated`, `ContrastsModeratedDEqMS`, `ContrastsPlotter`, `ContrastsRLMFacade`, `ContrastsROPECA`, `ContrastsROPECANestedFacade`, `ContrastsRfitFacade`, `ContrastsRfitImputeFacade`, `ContrastsTable`, `INTERNAL_FUNCTIONS_BY_FAMILY`, `LR_test()`, `Model`, `ModelFirth`, `ModelLimma`, `StrategyLM`, `StrategyLimma`, `StrategyLimpafacade`, `StrategyLmer`, `StrategyLogistf`, `StrategyRLM`, `StrategyRfit`, `build_contrast_analysis()`, `build_model()`, `build_model_glm_peptide()`, `build_model_glm_protein()`, `build_model_impute()`, `build_model_limma()`, `build_model_limma_impute()`, `build_model_limma_voom()`, `build_model_limma_voom_impute()`, `build_model_limpafacade()`, `build_model_logistf()`, `compute_borrowed_variance()`, `compute_borrowed_variance_contrast()`, `compute_lmer_contrast()`, `df.residual.rfit_prolfqua()`, `get_anova_df()`, `get_complete_model_fit()`, `get_p_values_pbeta()`, `group_label()`, `impute_from_model()`, `impute_refit_singular()`, `linfct_from_model()`, `linfct_matrix_contrasts()`, `list_facades()`, `lookup_facade()`, `merge_contrasts_results()`, `model_analyse()`, `model_summary()`, `moderated_p_deqms()`, `moderated_p_deqms_long()`, `moderated_p_limma()`, `new_imputed_model()`, `pivot_model_contrasts_to_wide()`, `register_facade()`, `sigma.rfit_prolfqua()`, `sim_build_models_lm()`, `sim_build_models_lmer()`, `sim_build_models_logistf()`, `sim_make_model_lm()`, `sim_make_model_lmer()`, `strategy_limma()`, `strategy_limpafacade()`, `summary_ROPECA_median_p_scaled()`, `vcov.rfit_prolfqua()`

Examples

```
tmp <- strategy_logistf("bin_resp ~ condition", model_name = "parallel design")
tmp$formula

istar <- prolfqua::sim_lfq_data_peptide_config(Nprot = 10, with_missing = TRUE,
  weight_missing = 0.5, seed = 3)
istar$data <- encode_bin_resp(LFQData$new(istar$data, istar$config))
istar$config$bin_resp <- "bin_resp"
istar <- LFQData$new(istar$data, istar$config)
df <- istar$summarize_hierarchy()
df2 <- df[df[[ncol(df)]] > 1, ]
```

```

istar2 <- istar$get_subset(df2)
istar2$data_long() |>
  dplyr::group_by(protein_Id) |>
  tidyr::nest() -> nestProtein
modelFunction <- strategy_logistf("bin_resp ~ group_ + peptide_Id",
  model_name = "random_example")
modelFunction$model_fun(nestProtein$data[[1]])
modelFunction$model_fun(nestProtein$data[[4]])
tmp <- strategy_rfit("Intensity ~ condition", model_name = "parallel design")
tmp$formula
modelFunction <- strategy_lmer("abundanceC ~ group_ + (1|peptide_Id)",
  model_name = "random_example")
modelFunction$formula
tmp <- strategy_lm("Intensity ~ condition", model_name = "parallel design")
tmp$formula
tmp$weights
tmp <- strategy_rlm("Intensity ~ condition", model_name = "parallel design")
tmp$formula

```

summarize_hierarchy	<i>Summarize hierarchy counts</i>
---------------------	-----------------------------------

Description

E.g compute number of peptides for each protein

Usage

```

summarize_hierarchy(
  pdata,
  hierarchy_keys,
  isotope_label = "isotopeLabel",
  hierarchy = hierarchy_keys,
  factors = character()
)

```

Arguments

pdata	data.frame
hierarchy_keys	character vector — all hierarchy column names
isotope_label	character — isotope label column name
hierarchy	character vector — hierarchy level to group by (default hierarchy_keys)
factors	character vector — factor columns to include

See Also

Other summary: [HierarchyCountsSample](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [hierarchy_counts\(\)](#)

Examples

```
bb <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb$data, bb$config)
summarize_hierarchy(lfq$data_long(), lfq$hierarchy_keys(), lfq$isotope_label())
summarize_hierarchy(lfq$data_long(), lfq$hierarchy_keys(), lfq$isotope_label(),
  factors = character())
summarize_hierarchy(lfq$data_long(), lfq$hierarchy_keys(), lfq$isotope_label(),
  hierarchy = lfq$relevant_hierarchy_keys())
```

summarize_stats	<i>Compute mean, sd, and CV for all Peptides, or proteins, for all interactions and all samples.</i>
-----------------	--

Description

Compute mean, sd, and CV for all Peptides, or proteins, for all interactions and all samples.
 summarize_stats output (compute quantiles)

Usage

```
summarize_stats(lfqdata, factor_key = lfqdata$relevant_factor_keys())

summarize_stats_quantiles(
  stats_res,
  factor_keys_depth,
  stats = c("sd", "CV"),
  probs = c(0.1, 0.25, 0.5, 0.75, 0.9)
)
```

Arguments

lfqdata	LFQData object
factor_key	character vector — factor columns to group by (default: relevant_factor_keys)
stats_res	result of running ‘summarize_stats’
factor_keys_depth	character vector — factor columns at current depth
stats	summarize either sd or CV
probs	for which quantiles 10, 20 etc.

See Also

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#)

Other stats: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [compute_pooled\(\)](#), [lfq_power_t_test_proteins\(\)](#), [lfq_power_t_test_quantiles_V2\(\)](#), [plot_stat_density\(\)](#), [plot_stat_density_median\(\)](#), [plot_stat_violin\(\)](#), [plot_stat_violin_median\(\)](#), [plot_stdv_vs_mean\(\)](#)

Examples

```

bb <- prolfqua::sim_lfq_data_protein_config()
lfq <- LFQData$new(bb$data, bb$config)
res1 <- summarize_stats(lfq)

res2 <- prolfqua::sim_lfq_data_2factor_config()
res2$config$factor_depth <- 2
lfq2 <- LFQData$new(res2$data, res2$config)
stats <- summarize_stats(lfq2)
stopifnot(nrow(stats) == 40)

stats <- summarize_stats(lfq2, factor_key = lfq2$factor_keys()[1])
stopifnot(nrow(stats) == 20)
stats <- summarize_stats(lfq2, factor_key = lfq2$factor_keys()[2])
stopifnot(nrow(stats) == 20)
stats <- summarize_stats(lfq2, factor_key = NULL)
stopifnot(nrow(stats) == 10)
library(ggplot2)
bb1 <- prolfqua::sim_lfq_data_peptide_config()
lfq <- LFQData$new(bb1$data, bb1$config)
stats_res <- summarize_stats(lfq)
sq <- summarize_stats_quantiles(stats_res, lfq$relevant_factor_keys())
sq <- summarize_stats_quantiles(stats_res, lfq$relevant_factor_keys(), stats = "CV")
sq <- summarize_stats_quantiles(stats_res, lfq$relevant_factor_keys(), stats = "sd")
xx <- summarize_stats_quantiles(stats_res, lfq$relevant_factor_keys(), probs = seq(0, 1, by = 0.1))
ggplot2::ggplot(xx$long, aes(x = probs, y = quantiles, color = group_)) + geom_line() + geom_point()

```

summarize_stats_factors

compute var sd etc for all factor levels

Description

compute var sd etc for all factor levels

Usage

```
summarize_stats_factors(lfqdata)
```

Arguments

lfqdata LFQData object

Value

The computed result.

Examples

```
res2 <- prolfqua::sim_lfq_data_2factor_config()
res2$config$factor_depth <- 2
lfq2 <- LFQData$new(res2$data, res2$config)
xx <- summarize_stats_factors(lfq2)
stopifnot(nrow(xx) == 80)
stopifnot(length(unique(xx$interaction)) == (2 + 2 + 2 * 2))
```

```
summary_ROPECA_median_p_scaled
```

compute protein level fold changes and p.values (using beta distribution) takes p-value of the scaled p-value

Description

compute protein level fold changes and p.values (using beta distribution) takes p-value of the scaled p-value

Usage

```
summary_ROPECA_median_p_scaled(
  contrasts_data,
  contrast = "contrast",
  subject_id = "protein_Id",
  estimate = "diff",
  statistic = "statistic",
  p.value = "moderated.p.value",
  max.n = 10
)
```

Arguments

contrasts_data	data frame
contrast	name of column with contrast identifier
subject_id	name of column with typically protein Id
estimate	name of column with effect size estimate
statistic	statistic name of column with statistic (typically t-statistics)
p.value	name of column with moderated.p.value
max.n	used to limit the number of peptides in probability computation.

Value

data.frame with columns

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df.residual.rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma.rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [vcov.rfit_prolfqua\(\)](#)

Examples

```
set.seed(10)
nrPep <- 10000
nrProtein <- 800
p.value <- runif(nrPep)
estimate <- rnorm(nrPep)
avgAbd <- runif(nrPep)
protein_Id <- sample(1:800, size = nrPep,
  replace = TRUE, prob = dexp(seq(0,5,length = 800)))

plot(table(table(protein_Id)))

testdata <- data.frame(contrast = "contrast1",
  protein_Id = protein_Id,
  estimate = estimate,
  pseudo_estimate = estimate,
  p.value = p.value,
  avgAbd = avgAbd )

xx30 <- summary_ROPECA_median_p_scaled(testdata,
  subject_id = "protein_Id",
  estimate = "estimate",
  p.value = "p.value",
  max.n = 30)

xx2 <- summary_ROPECA_median_p_scaled(testdata,
  subject_id = "protein_Id",
  estimate = "estimate",
  p.value = "p.value",
  max.n = 1)

testthat::expect_equal(mad(xx2$estimate, na.rm = TRUE),0.384409, tolerance = 1e-4)
```

```

testthat::expect_equal(median(xx2$estimate), -0.006874857, tolerance = 1e-4)
testthat::expect_equal(xx2$beta.based.significance[1], 0.819, tolerance = 1e-3)
testthat::expect_equal(xx2$beta.based.significance[2], 0.9234362, tolerance = 1e-3)

# Uniform distribution
hist(testdata$p.value)
hist(xx30$median.p.scaled, breaks = 20)
hist(xx2$median.p.scaled, breaks = 20)
# shows that beta.based.significance has NO uniform distribution
# although H0 is true for all cases.

hist(xx30$beta.based.significance, breaks = 20)
hist(xx2$beta.based.significance, breaks = 20)

hist(xx2$median.p.value, breaks = 20)
hist(xx2$beta.based.significance, breaks = 20)
hist(estimate)

```

table_facade	<i>table facade to easily switch implementations</i>
--------------	--

Description

table facade to easily switch implementations

Usage

```
table_facade(df, caption, digits = getOption("digits"), kable = TRUE)
```

Arguments

df	data.frame to display
caption	table caption
digits	number of digits (default from options)
kable	if TRUE use knitr::kable

Value

The requested plot, table, or transformed object.

See Also

Other utilities: [INTERNAL_FUNCTIONS_BY_FAMILY](#), [get_uniprot_id_from_fasta_header\(\)](#), [is_contaminant\(\)](#), [is_decoy\(\)](#), [matrix_to_tibble\(\)](#), [multigroup_volcano\(\)](#), [pairs_smooth\(\)](#), [panel_cor\(\)](#)

Examples

```
table_facade(head(iris, 3), caption = "Iris preview")
```

table_factors	<i>Table of distinct factors (sample annotation)</i>
---------------	--

Description

Table of distinct factors (sample annotation)

Usage

```
table_factors(pdata, file_name, sample_name, factor_keys)
```

Arguments

pdata	data.frame
file_name	character — file name column
sample_name	character — sample name column
factor_keys	character vector — factor column names

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_config\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors_size\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
xx <- table_factors(lfq$data_long(), lfq$file_name(), lfq$sample_name(), lfq$factor_keys())
xt <- xx |> dplyr::group_by(!!!rlang::syms(lfq$factor_keys())) |>
  dplyr::summarize(n = dplyr::n())
stopifnot(all(xt$n == 4))
```

table_factors_size	<i>Table of distinct factors with group sizes</i>
--------------------	---

Description

Table of distinct factors with group sizes

Usage

```
table_factors_size(
  pdata,
  file_name,
  sample_name,
  factor_keys,
  factor_keys_depth
)
```

Arguments

pdata	data.frame
file_name	character — file name column
sample_name	character — sample name column
factor_keys	character vector — all factor column names
factor_keys_depth	character vector — factor columns at current depth

See Also

Other configuration: [AnalysisConfiguration](#), [ContrastConfiguration](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [R6_extract_values\(\)](#), [complete_cases\(\)](#), [list_to_ContrastConfiguration\(\)](#), [make_interaction_column\(\)](#), [make_reduced_hierarchy_config\(\)](#), [sample_subset\(\)](#), [separate_hierarchy\(\)](#), [setup_analysis\(\)](#), [table_factors\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
xx <- table_factors_size(lfq$data_long(), lfq$file_name(),
  lfq$sample_name(), lfq$factor_keys(), lfq$relevant_factor_keys())
stopifnot(all(xx$n == 4))
```

tidy_to_wide_config	<i>transform long to wide</i>
---------------------	-------------------------------

Description

transform long to wide

Usage

```
tidy_to_wide_config(
  lfqdata,
  as.matrix = FALSE,
  file_name = FALSE,
  sep = "~lfq~",
  value = lfqdata$response()
)
```

Arguments

lfqdata	LFQData object
as.matrix	if TRUE return matrix, otherwise data.frame
file_name	if TRUE use file_name as column labels, otherwise sample_name
sep	separator for row IDs when as.matrix = TRUE
value	column name to pivot (default: lfqdata\$response())

Value

list with data, rowdata, and annotation (colData)

Examples

```
dd <- prolfqua::sim_lfq_data_peptide_config()
lfqdata <- prolfqua::LFQData$new(dd$data, dd$config)
res <- tidy_to_wide_config(lfqdata)
testthat::expect_equal(nrow(res$rowdata), nrow(res$data))
testthat::expect_equal(ncol(res$data) - ncol(res$rowdata) , nrow(res$annotation))
res <- tidy_to_wide_config(lfqdata, as.matrix = TRUE)
stopifnot(all(dim(res$data) == c(28, 12)))
stopifnot(all(dim(res$annotation) == c(12, 4)))
stopifnot(all(dim(res$rowdata) == c(28, 3)))

res <- scale(res$data)
tidy_to_wide_config(lfqdata, value = lfqdata$nr_children_col())

res <- lfqdata$get_Aggregator("medpolish")
x <- res$aggregate()
towide <- tidy_to_wide_config(x, value = x$nr_children_col())

dd <- prolfqua::sim_lfq_data_protein_config()
lfqprot <- prolfqua::LFQData$new(dd$data, dd$config)
xt <- tidy_to_wide_config(lfqprot, value = lfqprot$nr_children_col())
xt$data
```

transform_work_intensity

Transform intensity column by applying a function

Description

Transform intensity column by applying a function

Usage

```
transform_work_intensity(pdata, response, .func, .funcname = NULL)
```

Arguments

pdata	data.frame
response	character, name of the response column to transform
.func	function to transform intensities e.g. log2
.funcname	name of function (used for creating new column name)

Value

list with 'data' (data.frame) and 'colname' (new column name)

Examples

```

istar <- sim_lfq_data_peptide_config()
config <- istar$config
res <- transform_work_intensity(
  istar$data, config$get_response(), .func = log2
)
stopifnot("log2_abundance" %in% colnames(res$data))
res <- transform_work_intensity(
  istar$data, config$get_response(), .func = asinh
)
stopifnot("asinh_abundance" %in% colnames(res$data))

```

upset_interaction_missing_stats

UpSetR plot from interaction_missing_stats

Description

UpSetR plot from interaction_missing_stats

Usage

```
upset_interaction_missing_stats(lfqdata, tr = 2)
```

Arguments

lfqdata	LFQData object
tr	if less than tr observations in condition then missing

Value

The computed result.

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_missing_stats\(\)](#)

Other imputation: [get_contrast\(\)](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [upset_missing_stats\(\)](#)

Examples

```

istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
pups <- upset_interaction_missing_stats(lfq)
stopifnot(ncol(pups$data) == 5)
UpSetR::upset(pups$data, order.by = "freq", nsets = pups$nsets)

```

upset_missing_stats *prepare dataframe for UpSetR plot for all samples*

Description

prepare dataframe for UpSetR plot for all samples

Usage

```
upset_missing_stats(lfqdata)
```

Arguments

lfqdata LFQData object

See Also

Other plotting: [ContrastsPlotter](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [plot_estimate\(\)](#), [plot_heatmap\(\)](#), [plot_heatmap_cor\(\)](#), [plot_hierarchies_boxplot_df\(\)](#), [plot_hierarchies_line\(\)](#), [plot_intensity_distribution_violin\(\)](#), [plot_na_heatmap\(\)](#), [plot_pca\(\)](#), [plot_raster\(\)](#), [plot_sample_correlation\(\)](#), [upset_interaction_missing_stats\(\)](#)

Other imputation: [get_contrast\(\)](#), [missigness_histogram\(\)](#), [missingness_per_condition\(\)](#), [missingness_per_condition_cumsum\(\)](#), [upset_interaction_missing_stats\(\)](#)

Examples

```
istar <- sim_lfq_data_peptide_config()
lfq <- LFQData$new(istar$data, istar$config)
pups <- upset_missing_stats(lfq)
UpSetR::upset(pups$data, order.by = "freq", nsets = pups$nsets)
```

vcov.rfit_prolfqua *Named variance-covariance matrix for an augmented rfit fit*

Description

'Rfit::vcov.rfit' returns an unnamed matrix; 'compute_contrast()' indexes the covariance by coefficient name, so this method restores the dimnames from 'coef()'.

Usage

```
## S3 method for class 'rfit_prolfqua'
vcov(object, ...)
```

Arguments

object an 'rfit_prolfqua' fit
... passed to the next method

Value

the rfit covariance matrix with row/column names set to the coefficient names.

See Also

Other modelling: [AnovaExtractor](#), [Contrasts](#), [ContrastsDEqMSFacade](#), [ContrastsDEqMSVoomFacade](#), [ContrastsFacadeBase](#), [ContrastsFirth](#), [ContrastsFirthFacade](#), [ContrastsFirthNestedFacade](#), [ContrastsLMFacade](#), [ContrastsLMImputeFacade](#), [ContrastsLimma](#), [ContrastsLimmaFacade](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL_FUNCTIONS_BY_FAMILY](#), [LR_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [StrategyLimma](#), [StrategyLimpafacade](#), [StrategyLmer](#), [StrategyLogistf](#), [StrategyRLM](#), [StrategyRfit](#), [build_contrast_analysis\(\)](#), [build_model\(\)](#), [build_model_glm_peptide\(\)](#), [build_model_glm_protein\(\)](#), [build_model_impute\(\)](#), [build_model_limma\(\)](#), [build_model_limma_impute\(\)](#), [build_model_limma_voom\(\)](#), [build_model_limma_voom_impute\(\)](#), [build_model_limpafacade\(\)](#), [build_model_logistf\(\)](#), [compute_borrowed_variance\(\)](#), [compute_borrowed_variance_contrast\(\)](#), [compute_lmer_contrast\(\)](#), [df_residual_rfit_prolfqua\(\)](#), [get_anova_df\(\)](#), [get_complete_model_fit\(\)](#), [get_p_values_pbeta\(\)](#), [group_label\(\)](#), [impute_from_model\(\)](#), [impute_refit_singular\(\)](#), [linfct_from_model\(\)](#), [linfct_matrix_contrasts\(\)](#), [list_facades\(\)](#), [lookup_facade\(\)](#), [merge_contrasts_results\(\)](#), [model_analyse\(\)](#), [model_summary\(\)](#), [moderated_p_deqms\(\)](#), [moderated_p_deqms_long\(\)](#), [moderated_p_limma\(\)](#), [new_imputed_model\(\)](#), [pivot_model_contrasts_to_wide\(\)](#), [register_facade\(\)](#), [sigma_rfit_prolfqua\(\)](#), [sim_build_models_lm\(\)](#), [sim_build_models_lmer\(\)](#), [sim_build_models_logistf\(\)](#), [sim_make_model_lm\(\)](#), [sim_make_model_lmer\(\)](#), [strategy_limma\(\)](#), [strategy_limpafacade\(\)](#), [strategy_logistf\(\)](#), [summary_ROPECA_median_p_scaled\(\)](#)

Examples

```
fit <- strategy_rfit("Sepal.Length ~ Species")$model_fun(iris)
vcov(fit)
```

volcano_plotly	<i>volcano plotly</i>
----------------	-----------------------

Description

volcano plotly

Usage

```
volcano_plotly(
  .data,
  effect = "fc",
  significance = "BFDR",
  contrast = "condition",
  proteinID = "Prey",
  color = "modelName",
  palette = NULL,
  xintercept = c(-2, 2),
  yintercept = 0.1,
  minsignificance = NULL,
```

```
    title_size = 25,  
    group = "BB"  
  )
```

Arguments

.data	data frame
effect	effect size x-axis
significance	significance
contrast	column with contrast labels
proteinID	column with protein ids
color	column used for colouring points
palette	named colour vector for the colour aesthetic
xintercept	vertical abline at x
yintercept	horizontal abline at y
minsignificance	optional minimum significance value (floor for -log10 axis). If NULL, only exact zero and negative values are replaced with the smallest positive observed value in the same score column.
title_size	font size of the subplot title annotation
group	crosstalk group name for linked brushing

Value

The requested plot, table, or transformed object.

Examples

```
data <- data.frame(fc = c(-1,0,1,2,8), BFDR = c(0.01,1, 0.01, 0.005,0),  
condition = rep("A",5), Prey = LETTERS[1:5], modelName = c("A","A","B","A","A"))  
  
dataB <- data.frame(fc = c(-1,0,1,2,8), BFDR = c(0.01,1, 0.01, 0.005,0),  
condition = rep("B",5), Prey = LETTERS[1:5],modelName = c("A","A","B","B","B"))  
data <- dplyr::bind_rows(data, dataB)  
bc <- volcano_plotly(data, xintercept = 1, yintercept= 0.01, palette = c(A = "black" , B = "red"))  
bc |> plotly::subplot()
```

which_missing	<i>add missing values to x vector based on the values of x</i>
---------------	--

Description

add missing values to x vector based on the values of x

Usage

```
which_missing(x, weight_missing = 0.2)
```

Arguments

x vector of intensities
weight_missing greater weight more missing

Value

A logical vector indicating which values are missing.

Examples

which_missing(2**rnorm(10,2,0.4))

x5463yzwer453bbb	<i>Factor annotation of 36 samples</i>
------------------	--

Description

Sample annotation table with two factors ('factor_A': MI / MINOCA, 'factor_B': sampling time T0 / T150 / T300), used to demonstrate how contrasts are derived from the factor levels. See 'inst/script/README.md' for its origin.

Usage

x5463yzwer453bbb

Format

A tibble with 36 rows and the columns 'Relative Path', 'Name', 'Group', 'factor_A', 'factor_B'.

See Also

Other data: [PACKAGE_DATA](#), [data_checksummarizationrobust87](#), [data_checksummarizerobust69](#), [data_ionstar](#), [prolfqua_data\(\)](#)

Index

* LFQData

- AggregateLimpa, [10](#)
- AggregateMedpolish, [12](#)
- AggregateRlm, [13](#)
- AggregateTopN, [14](#)
- AggregatorBase, [16](#)
- LFQData, [131](#)
- LFQDataPlotter, [137](#)
- LFQDataStats, [142](#)
- LFQDataSummariser, [146](#)
- LFQDataToSummarizedExperiment, [149](#)

* aggregation

- aggregate_intensity_top_n, [15](#)
- estimate_intensity, [112](#)
- INTERNAL_FUNCTIONS_BY_FAMILY, [128](#)
- plot_estimate, [194](#)
- plot_hierarchies_line, [199](#)

* configuration

- AnalysisConfiguration, [18](#)
- complete_cases, [39](#)
- ContrastConfiguration, [45](#)
- INTERNAL_FUNCTIONS_BY_FAMILY, [128](#)
- list_to_ContrastConfiguration, [159](#)
- make_interaction_column, [163](#)
- make_reduced_hierarchy_config, [164](#)
- R6_extract_values, [209](#)
- sample_subset, [213](#)
- separate_hierarchy, [215](#)
- setup_analysis, [216](#)
- table_factors, [253](#)
- table_factors_size, [253](#)

* contrasts

- annotation_add_contrasts, [21](#)
- generate_contrasts, [115](#)
- generate_contrasts_for_factor, [116](#)
- interaction_contrasts, [127](#)
- level_specific_contrasts, [130](#)
- main_effect_contrasts, [162](#)

* datasets

- FACADE_REGISTRY, [113](#)

* data

- data_checksummarizationrobust87, [109](#)

- data_checksummarizerobust69, [109](#)

- data_ionstar, [110](#)

- PACKAGE_DATA, [191](#)

- prolfqua_data, [209](#)

- x5463yzwer453bbb, [260](#)

* imputation

- get_contrast, [118](#)
- missigness_histogram, [166](#)
- missingness_per_condition, [169](#)
- missingness_per_condition_cumsum, [169](#)
- upset_interaction_missing_stats, [256](#)
- upset_missing_stats, [257](#)

* internal

- .get_robscales, [7](#)
- .lfqdata_to_elist, [7](#)
- .make_progress, [8](#)
- .resolve_weights, [8](#)
- aggregate_intensity_top_n, [15](#)
- AggregatorBase, [16](#)
- apply_to_response_matrix, [23](#)
- build_model_logistf, [38](#)
- complete_cases, [39](#)
- compute_borrowed_variance, [40](#)
- compute_borrowed_variance_limma, [41](#)
- compute_contrast, [42](#)
- compute_lmer_contrast, [43](#)
- compute_pooled, [44](#)
- contrasts_linfct, [108](#)
- ContrastsFacadeBase, [55](#)
- data_checksummarizationrobust87, [109](#)
- data_checksummarizerobust69, [109](#)
- data_ionstar, [110](#)
- df.residual.rlm, [111](#)
- encode_bin_resp, [112](#)
- estimate_intensity, [112](#)
- filter_proteins_by_peptide_count, [114](#)
- find_d0_deqms, [114](#)
- get_anova_df, [117](#)

- get_complete_model_fit, 117
- get_contrast, 118
- get_p_values_pbeta, 119
- get_robustscales, 120
- hierarchy_counts, 124
- impute_refit_singular, 126
- lfq_power_t_test_proteins, 153
- lfq_power_t_test_quantiles_V2, 154
- linfct_from_model, 155
- linfct_matrix_contrasts, 156
- make_interaction_column, 163
- make_reduced_hierarchy_config, 164
- missigness_histogram, 166
- missingness_per_condition, 169
- missingness_per_condition_cumsum, 169
- model_analyse, 179
- model_summary, 180
- moderated_p_deqms, 181
- moderated_p_deqms_long, 182
- moderated_p_limma, 183
- multigroup_volcano, 184
- new_imputed_model, 186
- nr_B_in_A, 187
- pairs_smooth, 191
- panel_cor, 192
- plot_estimate, 194
- plot_heatmap, 194
- plot_heatmap_cor, 196
- plot_hierarchies_boxplot, 197
- plot_hierarchies_boxplot_df, 198
- plot_hierarchies_line, 199
- plot_intensity_distribution_violin, 200
- plot_lrt_diagnostics, 201
- plot_na_heatmap, 201
- plot_pca, 202
- plot_raster, 203
- plot_sample_correlation, 204
- plot_stat_density, 205
- plot_stat_density_median, 206
- plot_stat_violin, 206
- plot_stat_violin_median, 207
- plot_stdv_vs_mean, 208
- prolfqua-package, 6
- rank_peptide_by_intensity, 210
- response_matrix_as_tibble, 212
- robust_scale, 213
- sample_subset, 213
- scale_with_subset, 214
- script_copy_helper_vec, 215
- separate_hierarchy, 215
- sigma.rlm, 218
- sim_build_models_lm, 219
- sim_build_models_lmer, 220
- sim_build_models_logistf, 221
- sim_make_model_lm, 225
- sim_make_model_lmer, 226
- StrategyBase, 227
- summarize_hierarchy, 247
- summarize_stats, 248
- summary_ROPECA_median_p_scaled, 250
- table_factors, 253
- table_factors_size, 253
- tidy_to_wide_config, 254
- transform_work_intensity, 255
- upset_missing_stats, 257
- x5463yzwer453bbb, 260
- * modelling**
 - AnovaExtractor, 22
 - build_contrast_analysis, 24
 - build_model, 26
 - build_model_glm_peptide, 27
 - build_model_glm_protein, 28
 - build_model_impute, 29
 - build_model_limma, 31
 - build_model_limma_impute, 32
 - build_model_limma_voom, 33
 - build_model_limma_voom_impute, 35
 - build_model_limpa, 36
 - build_model_logistf, 38
 - compute_borrowed_variance, 40
 - compute_borrowed_variance_limma, 41
 - compute_contrast, 42
 - compute_lmer_contrast, 43
 - Contrasts, 48
 - ContrastsDEqMSFacade, 52
 - ContrastsDEqMSVoomFacade, 54
 - ContrastsFacadeBase, 55
 - ContrastsFirth, 58
 - ContrastsFirthFacade, 60
 - ContrastsFirthNestedFacade, 61
 - ContrastsLimma, 66
 - ContrastsLimmaFacade, 69
 - ContrastsLimmaImputeFacade, 70
 - ContrastsLimmaVoomFacade, 72
 - ContrastsLimmaVoomImputeFacade, 74
 - ContrastsLimpafacade, 76
 - ContrastsLimpaNestedFacade, 78
 - ContrastsLmerNestedFacade, 80
 - ContrastsLMFacade, 81
 - ContrastsLMImputeFacade, 83

- ContrastsModerated, 85
- ContrastsModeratedDEqMS, 87
- ContrastsPlotter, 90
- ContrastsRfitFacade, 95
- ContrastsRfitImputeFacade, 97
- ContrastsRLMFacade, 99
- ContrastsROPECA, 100
- ContrastsROPECANestedFacade, 103
- ContrastsTable, 106
- df.residual.rfit_prolfqua, 110
- get_anova_df, 117
- get_complete_model_fit, 117
- get_p_values_pbeta, 119
- group_label, 121
- impute_from_model, 124
- impute_refit_singular, 126
- INTERNAL_FUNCTIONS_BY_FAMILY, 128
- linfct_from_model, 155
- linfct_matrix_contrasts, 156
- list_facades, 157
- lookup_facade, 160
- LR_test, 161
- merge_contrasts_results, 165
- Model, 170
- model_analyse, 179
- model_summary, 180
- ModelFirth, 172
- ModelLimma, 176
- moderated_p_deqms, 181
- moderated_p_deqms_long, 182
- moderated_p_limma, 183
- new_imputed_model, 186
- pivot_model_contrasts_to_wide, 192
- register_facade, 211
- sigma.rfit_prolfqua, 217
- sim_build_models_lm, 219
- sim_build_models_lmer, 220
- sim_build_models_logistf, 221
- sim_make_model_lm, 225
- sim_make_model_lmer, 226
- strategy_limma, 241
- strategy_limpa, 242
- strategy_logistf, 244
- StrategyLimma, 230
- StrategyLimpa, 232
- StrategyLM, 233
- StrategyLmer, 235
- StrategyLogistf, 236
- StrategyRfit, 238
- StrategyRLM, 239
- summary_ROPECA_median_p_scaled, 250
- vcov.rfit_prolfqua, 257
- * plotting**
 - ContrastsPlotter, 90
 - INTERNAL_FUNCTIONS_BY_FAMILY, 128
 - missigness_histogram, 166
 - missingness_per_condition, 169
 - missingness_per_condition_cumsum, 169
 - plot_estimate, 194
 - plot_heatmap, 194
 - plot_heatmap_cor, 196
 - plot_hierarchies_boxplot_df, 198
 - plot_hierarchies_line, 199
 - plot_intensity_distribution_violin, 200
 - plot_na_heatmap, 201
 - plot_pca, 202
 - plot_raster, 203
 - plot_sample_correlation, 204
 - upset_interaction_missing_stats, 256
 - upset_missing_stats, 257
- * preprocessing**
 - apply_to_response_matrix, 23
 - filter_proteins_by_peptide_count, 114
 - get_robscales, 120
 - INTERNAL_FUNCTIONS_BY_FAMILY, 128
 - nr_B_in_A, 187
 - robust_scale, 213
 - scale_with_subset, 214
- * stats**
 - compute_pooled, 44
 - INTERNAL_FUNCTIONS_BY_FAMILY, 128
 - lfq_power_t_test_proteins, 153
 - lfq_power_t_test_quantiles_V2, 154
 - plot_stat_density, 205
 - plot_stat_density_median, 206
 - plot_stat_violin, 206
 - plot_stat_violin_median, 207
 - plot_stdv_vs_mean, 208
 - summarize_stats, 248
- * summary**
 - hierarchy_counts, 124
 - HierarchyCountsSample, 122
 - INTERNAL_FUNCTIONS_BY_FAMILY, 128
 - summarize_hierarchy, 247
- * utilities**
 - get_uniprot_id_from_fasta_header, 121
 - INTERNAL_FUNCTIONS_BY_FAMILY, 128
 - is_contaminant, 129

- is_decoy, 130
- matrix_to_tibble, 164
- multigroup_volcano, 184
- pairs_smooth, 191
- panel_cor, 192
- table_facade, 252
- .get_robustscales, 7
- .lfqdata_to_elist, 7
- .make_progress, 8
- .resolve_weights, 8
- adjust_p_values, 9
- aggregate_intensity_top_n, 15, 112, 128, 194, 199
- AggregateLimpa, 10, 12, 14–16, 18, 36, 76–79, 137, 139, 142, 145, 148, 149
- AggregateMedpolish, 11, 12, 14–16, 18, 137, 142, 145, 148, 149
- AggregateRlm, 11, 12, 13, 15, 16, 18, 137, 142, 145, 148, 149
- AggregateTopN, 11, 12, 14, 14, 16, 18, 137, 142, 145, 148, 149
- AggregatorBase, 11, 12, 14, 15, 16, 137, 142, 145, 148, 149
- AnalysisConfiguration, 18, 40, 45, 47, 110, 128, 159, 163, 164, 210, 213, 216, 253, 254
- annotation_add_contrasts, 21, 116, 127, 131, 163
- AnovaExtractor, 22, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- apply_to_response_matrix, 23, 114, 120, 129, 188, 213, 214
- build_contrast_analysis, 23, 24, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242, 243, 245, 246, 251, 258
- build_model, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 52, 53, 55, 57–59, 61, 62, 68, 70, 72, 73, 75, 77, 79–84, 86, 89, 94–96, 98–100, 102, 104, 105, 107, 111, 117–119, 122, 124, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242, 243, 245, 246, 251, 258
- build_model_glm_peptide, 23, 25, 27, 27, 29–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242, 243, 245, 246, 251, 258
- build_model_glm_protein, 23, 25, 27, 28, 28, 30, 31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59–62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 172, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242, 243, 245, 246, 251, 258
- build_model_impute, 23, 25, 27–29, 29, 31–34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96–98, 100, 102, 105, 107, 111, 117–119, 122, 124, 125, 127, 128, 155, 157, 158, 160, 161, 166, 167, 172, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242, 243, 245, 246, 251, 258
- build_model_limma, 23, 25, 27–30, 31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68–70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 172, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227,

- 230, 231, 233, 234, 236, 237, 239,
240, 242, 243, 245, 246, 251, 258
- build_model_limma_impute, 23, 25, 27–31,
32, 34–38, 41–44, 49, 53, 55, 57, 59,
61, 62, 68, 70–73, 75, 77, 79, 81, 82,
84, 86, 89, 94, 96, 98, 100, 102, 105,
107, 111, 117–119, 122, 125, 127,
128, 155, 157, 158, 160, 161, 166,
172, 174, 178–180, 182–184, 186,
193, 212, 218–221, 226, 227, 231,
233, 234, 236, 237, 239, 240, 242,
243, 245, 246, 251, 258
- build_model_limma_voom, 23, 25, 27–31, 33,
33, 36–38, 41–44, 49, 53–55, 57, 59,
61, 62, 68, 70, 72, 73, 75, 77, 79, 81,
82, 84, 86, 89, 94, 96, 98, 100, 102,
105, 107, 111, 117–119, 122, 125,
127, 128, 155, 157, 158, 160, 161,
166, 172, 174, 178–180, 182–184,
186, 193, 212, 218–221, 226, 227,
231, 233, 234, 236, 238–240, 242,
243, 245, 246, 251, 258
- build_model_limma_voom_impute, 23, 25,
27–31, 33, 34, 35, 37, 38, 41–44, 49,
53, 55, 57, 59, 61, 62, 68, 70, 72–75,
77, 79, 81, 82, 84, 86, 89, 94, 96, 98,
100, 102, 105, 107, 111, 117–119,
122, 125, 127, 128, 155, 157, 158,
160, 161, 166, 172, 174, 178–180,
182–184, 186, 193, 212, 218–221,
226, 227, 231, 233, 234, 236,
238–240, 242, 243, 245, 246, 251,
258
- build_model_limpa, 23, 25, 27–31, 33, 34,
36, 36, 38, 41–44, 49, 53, 55, 57, 59,
61, 62, 68, 70, 72, 74–79, 81, 82, 84,
86, 89, 94, 96, 98, 100, 102, 105,
107, 111, 117–119, 122, 125, 127,
128, 155, 157, 158, 160, 161, 166,
172, 174, 178–180, 182–184, 186,
193, 212, 218–221, 226, 227,
231–234, 236, 238–240, 242, 243,
245, 246, 251, 258
- build_model_logistf, 23, 25, 27–31, 33, 34,
36, 37, 38, 41–44, 49, 53, 55, 57, 59,
61, 62, 68, 70, 72, 74, 75, 77, 79, 81,
82, 84, 86, 89, 94, 96, 98, 100, 102,
105, 107, 111, 117–119, 122, 125,
127, 128, 155, 157, 158, 160, 161,
166, 172, 174, 178–180, 182–184,
186, 193, 212, 218–221, 226, 227,
231, 233, 234, 236, 238–240, 242,
243, 245, 246, 251, 258
- 243, 245, 246, 251, 258
- center_to_reference_cfg, 39
- complete_cases, 20, 39, 47, 128, 159, 163,
164, 210, 213, 216, 253, 254
- compute_borrowed_variance, 23, 25, 27–31,
33, 34, 36–38, 40, 42–44, 49, 53, 55,
57, 59, 61, 62, 68, 70, 72, 74, 75, 77,
79, 81, 82, 84, 86, 89, 94, 96, 98,
100, 102, 105, 107, 111, 117–119,
122, 125–128, 155, 157, 158, 160,
162, 166, 172, 174, 178–180,
182–184, 186, 193, 212, 218–221,
226, 227, 231, 233, 234, 236,
238–240, 242, 243, 245, 246, 251,
258
- compute_borrowed_variance_limma, 23, 25,
27–31, 33, 34, 36–38, 41, 41, 43, 44,
49, 53, 55, 57, 59, 61, 62, 68, 70, 72,
74, 75, 77, 79, 81, 82, 84, 86, 89, 94,
96, 98, 100, 102, 105, 107, 111,
117–119, 122, 125, 127, 128, 155,
157, 158, 160, 162, 166, 172, 174,
178–180, 182–184, 186, 193, 212,
218–221, 226, 227, 231, 233, 234,
236, 238–240, 242, 243, 245, 246,
251, 258
- compute_contrast, 23, 25, 27–31, 33, 34,
36–38, 41, 42, 42, 44, 48, 49, 53, 55,
57, 59, 61, 62, 68, 70, 72, 74, 75, 77,
79, 81, 82, 84, 86, 89, 94, 96, 98,
100, 102, 105, 107, 111, 117–119,
122, 125, 127, 128, 155, 157, 158,
160, 162, 166, 172, 174, 178–180,
182–184, 186, 193, 212, 218–221,
226–228, 231, 233, 234, 236,
238–240, 242, 243, 245, 246, 251,
258
- compute_lmer_contrast, 23, 25, 27–31, 33,
34, 36–38, 41–43, 43, 49, 53, 55, 57,
59, 61, 62, 68, 70, 72, 74, 75, 77, 79,
81, 82, 84, 86, 89, 94, 96, 98, 100,
102, 105, 107, 111, 117–119, 122,
125, 127, 128, 155, 157, 158, 160,
162, 166, 172, 174, 178–180,
182–184, 186, 193, 212, 218–221,
226, 227, 231, 233–236, 238–240,
242, 243, 245, 246, 251, 258
- compute_pooled, 44, 129, 154, 155, 205–208,
248
- ContrastConfiguration, 20, 40, 45, 56,
63–65, 128, 159, 163, 164, 210, 213,
216, 253, 254

- Contrasts, 23, 25, 26, 28–31, 33, 34, 36–38, 41–45, 48, 52, 53, 55, 57, 59, 61, 62, 66, 68, 70, 72, 73, 75, 77, 79–84, 86, 89, 94–100, 102, 104, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 165, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- contrasts_linfct, 108
- ContrastsBinomialNestedFacade, 25, 51
- ContrastsDEqMSFacade, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 52, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsDEqMSVoomFacade, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 54, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsFacadeBase, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 55, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsFirth, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 58, 60–62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsFirthFacade, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 60, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsFirthNestedFacade, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 51, 53, 55, 57, 59–61, 61, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsInterface, 63, 66
- ContrastsLimma, 23, 25, 26, 28–31, 33, 34, 36–38, 41–45, 49, 53–55, 57, 59, 61, 62, 66, 69–79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsLimmaFacade, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 69, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ContrastsLimmaImputeFacade, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 70, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–181, 183, 184, 186, 193, 211, 217, 219–221, 226, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258

- ContrastsLimmaVoomFacade, 23, 25, 26,
28–31, 33, 34, 36–38, 41–44, 49, 53,
55, 57, 59, 61, 62, 68, 70, 72, 72, 75,
77, 79, 81, 82, 84, 86, 89, 94, 96, 98,
100, 102, 105, 107, 111, 117–119,
122, 125, 127, 128, 155, 157, 158,
160, 161, 166, 171, 174, 178–181,
183, 184, 186, 193, 211, 217,
219–221, 226, 231, 233, 234, 236,
237, 239, 240, 242–246, 251, 258
- ContrastsLimmaVoomImputeFacade, 23, 25,
26, 28–31, 33, 34, 36–38, 41–44, 49,
53, 55, 57, 59, 61, 62, 68, 70, 72, 73,
74, 77, 79, 81, 82, 84, 86, 89, 94, 96,
98, 100, 102, 105, 107, 111,
117–119, 122, 125, 127, 128, 155,
157, 158, 160, 161, 166, 171, 174,
178–181, 183, 184, 186, 193, 211,
217, 219–221, 226, 231, 233, 234,
236, 237, 239, 240, 242–246, 251,
258
- ContrastsLimpaFacade, 23, 25, 26, 28–31,
33, 34, 36–38, 41–44, 49, 53, 55, 57,
59, 61, 62, 68, 70, 72, 73, 75, 76, 78,
79, 81, 82, 84, 86, 89, 94, 96, 98,
100, 102, 105, 107, 111, 117–119,
122, 125, 127, 128, 155, 157, 158,
160, 161, 166, 171, 174, 178–181,
183, 184, 186, 193, 211, 217,
219–221, 226, 231, 233, 234, 236,
237, 239, 240, 242–246, 251, 258
- ContrastsLimpaNestedFacade, 23, 25, 26,
28–31, 33, 34, 36–38, 41–44, 49, 53,
55, 57, 59, 61, 62, 68, 70, 72, 73,
75–77, 78, 81, 82, 84, 86, 89, 94, 96,
98, 100, 102, 105, 107, 111,
117–119, 122, 125, 127, 128, 155,
157, 158, 160, 161, 166, 171, 174,
178–181, 183, 184, 186, 193, 211,
217, 219–221, 226, 231, 233, 234,
236, 237, 239, 240, 242–246, 251,
258
- ContrastsLmerNestedFacade, 23, 25, 26,
28–31, 33, 34, 36–38, 41–44, 49, 53,
55, 57, 59, 61, 62, 68, 70, 72, 73, 75,
77, 79, 80, 82, 84, 86, 89, 94, 96, 98,
100, 102, 105, 107, 111, 117–119,
122, 125, 127, 128, 155, 157, 158,
160, 161, 166, 171, 174, 178–181,
183, 184, 186, 193, 211, 218–221,
226, 231, 233, 234, 236, 237, 239,
240, 242–246, 251, 258
- ContrastsLMFacade, 23, 25, 26, 28–31, 33,
34, 36–38, 41–44, 49, 53, 55, 57, 59,
61, 62, 68, 70, 72, 73, 75, 77, 79, 81,
81, 84, 86, 89, 94–96, 98, 100, 102,
105, 107, 111, 117–119, 122, 125,
127, 128, 155, 157, 158, 160, 161,
166, 171, 174, 178–181, 183, 184,
186, 193, 211, 217, 219–221, 226,
231, 233, 234, 236, 237, 239, 240,
242–246, 251, 258
- ContrastsLMImputeFacade, 23, 25, 26,
28–31, 33, 34, 36–38, 41–44, 49, 53,
55, 57, 59, 61, 62, 68, 70, 72, 73, 75,
77, 79, 81, 82, 83, 86, 89, 94, 96–98,
100, 102, 105, 107, 111, 117–119,
122, 125, 127, 128, 155, 157, 158,
160, 161, 165, 166, 171, 174,
178–181, 183, 184, 186, 193, 211,
217, 219–221, 226, 231, 233, 234,
236, 237, 239, 240, 242–246, 251,
258
- ContrastsModerated, 23, 25, 26, 28–31, 33,
34, 36–38, 41–45, 49, 51, 53, 55, 57,
59, 61, 62, 68, 70, 72, 73, 75, 77,
79–84, 85, 88, 89, 94–100, 102, 105,
107, 111, 117–119, 122, 125, 127,
128, 155, 157, 158, 160, 161, 166,
171, 174, 178–181, 183, 184, 186,
193, 211, 218–221, 226, 231, 233,
234, 236, 237, 239, 240, 242–246,
251, 258
- ContrastsModeratedDEqMS, 23, 25, 26,
28–31, 33, 34, 36–38, 41–44, 49,
52–55, 57, 59, 61, 62, 66–68, 70, 72,
73, 75, 77, 79, 81, 82, 84, 86, 87, 94,
96, 98, 100, 102, 105, 107, 111,
117–119, 122, 125, 127, 128, 155,
157, 158, 160, 161, 166, 171, 174,
178–181, 183, 184, 186, 193, 211,
218–221, 226, 231, 233, 234, 236,
237, 239, 240, 242–246, 251, 258
- ContrastsPlotter, 23, 25, 26, 28–31, 33, 34,
36–38, 41–44, 49, 53, 55, 57, 59, 61,
62, 64, 68, 70, 72, 73, 75, 77, 79, 81,
82, 84, 86, 89, 90, 96, 98, 100, 102,
105, 107, 111, 117–119, 122, 125,
127, 128, 155, 157, 158, 160, 161,
166, 167, 169–171, 174, 178–181,
183, 184, 186, 193–196, 198–200,
202–205, 211, 218–221, 226, 231,
233, 234, 236, 237, 239, 240,
242–246, 251, 256–258

- ContrastsRfitFacade, [23](#), [25](#), [26](#), [28–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [73](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [95](#), [98](#), [100](#), [102](#), [105](#), [107](#), [111](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [161](#), [166](#), [171](#), [174](#), [178–181](#), [183](#), [184](#), [186](#), [193](#), [211](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [237](#), [239](#), [240](#), [242–246](#), [251](#), [258](#)
- ContrastsRfitImputeFacade, [23](#), [25](#), [26](#), [28–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [73](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [97](#), [100](#), [102](#), [105](#), [107](#), [111](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [161](#), [166](#), [171](#), [174](#), [178–180](#), [182–184](#), [186](#), [193](#), [211](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [237](#), [239](#), [240](#), [242–246](#), [251](#), [258](#)
- ContrastsRLMFacade, [23](#), [25](#), [26](#), [28–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [73](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [98](#), [99](#), [102](#), [105](#), [107](#), [111](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [161](#), [166](#), [171](#), [174](#), [178–181](#), [183](#), [184](#), [186](#), [193](#), [211](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [237](#), [239](#), [240](#), [242–246](#), [251](#), [258](#)
- ContrastsROPECA, [23](#), [25](#), [26](#), [28–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [73](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [98](#), [100](#), [100](#), [104](#), [105](#), [107](#), [111](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [161](#), [166](#), [171](#), [174](#), [178–181](#), [183](#), [184](#), [186](#), [193](#), [211](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [237](#), [239](#), [240](#), [242–246](#), [251](#), [258](#)
- ContrastsROPECANestedFacade, [23](#), [25](#), [26](#), [28–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [73](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [98](#), [100](#), [102](#), [103](#), [107](#), [111](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [161](#), [166](#), [171](#), [174](#), [178–181](#), [183](#), [184](#), [186](#), [193](#), [211](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [237](#), [239](#), [240](#), [242–246](#), [251](#), [258](#)
- ContrastsTable, [23](#), [25](#), [26](#), [28–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [73](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [98](#), [100](#), [102](#), [105](#), [107](#), [111](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [161](#), [166](#), [171](#), [174](#), [178–180](#), [182–184](#), [186](#), [193](#), [211](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [237](#), [239](#), [240](#), [242–246](#), [251](#), [258](#)
- data_checksummarizationrobust87, [109](#), [109](#), [110](#), [191](#), [209](#), [260](#)
- data_checksummarizerobust69, [109](#), [109](#), [110](#), [191](#), [209](#), [260](#)
- data_ionstar, [109](#), [110](#), [191](#), [209](#), [260](#)
- df.residual.rfit_prolfqua, [23](#), [25](#), [27–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [74](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [98](#), [100](#), [102](#), [105](#), [107](#), [110](#), [117–119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [162](#), [166](#), [172](#), [174](#), [178–180](#), [182–184](#), [186](#), [193](#), [212](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [238–240](#), [242](#), [243](#), [245](#), [246](#), [251](#), [258](#)
- df.residual.rlm, [111](#)
- eBayes, [230](#), [232](#), [242](#), [243](#)
- encode_bin_resp, [112](#)
- estimate_intensity, [16](#), [112](#), [128](#), [194](#), [199](#)
- FACADE_REGISTRY, [113](#), [158](#)
- facet_wrap, [92](#), [93](#)
- filter_proteins_by_peptide_count, [24](#), [114](#), [120](#), [129](#), [188](#), [213](#), [214](#)
- find_d0_deqms, [114](#)
- find_package_file, [115](#)
- generate_contrasts, [21](#), [115](#), [116](#), [127](#), [131](#), [163](#)
- generate_contrasts_for_factor, [21](#), [116](#), [116](#), [127](#), [131](#), [163](#)
- get_anova_df, [23](#), [25](#), [27–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#), [61](#), [62](#), [68](#), [70](#), [72](#), [74](#), [75](#), [77](#), [79](#), [81](#), [82](#), [84](#), [86](#), [89](#), [94](#), [96](#), [98](#), [100](#), [102](#), [105](#), [107](#), [111](#), [117](#), [118](#), [119](#), [122](#), [125](#), [127](#), [128](#), [155](#), [157](#), [158](#), [160](#), [162](#), [166](#), [172](#), [174](#), [178–180](#), [182–184](#), [186](#), [193](#), [212](#), [218–221](#), [226](#), [227](#), [231](#), [233](#), [234](#), [236](#), [238–240](#), [242](#), [243](#), [245](#), [246](#), [251](#), [258](#)
- get_complete_model_fit, [23](#), [25](#), [27–31](#), [33](#), [34](#), [36–38](#), [41–44](#), [49](#), [53](#), [55](#), [57](#), [59](#),

- 61, 62, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117, 119, 122, 125, 127, 128, 155, 157, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- get_contrast, 118, 167, 169, 170, 256, 257
- get_p_values_pbeta, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117, 118, 119, 122, 125, 127, 128, 155, 157, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- get_rob scales, 24, 114, 120, 129, 188, 213, 214
- get_uniprot_id_from_fasta_header, 121, 129, 130, 165, 185, 191, 192, 252
- glm, 229
- group_label, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 121, 125, 127, 128, 155, 157, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- hierarchy_counts, 123, 124, 129, 247
- HierarchyCountsSample, 122, 124, 129, 247
- impute_from_model, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 124, 127, 128, 155, 157, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- impute_refit_singular, 23, 25, 27–31, 33, 34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- interaction_contrasts, 21, 116, 127, 131, 163
- INTERNAL_FUNCTIONS_BY_FAMILY, 16, 20, 23–26, 28–31, 33, 34, 36–38, 40–45, 47, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 112, 114, 117–125, 127, 128, 129, 130, 154, 155, 157–161, 163–167, 169–171, 174, 178–180, 182–186, 188, 191–196, 198–200, 202–208, 210, 211, 213, 214, 216, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242–248, 251–254, 256–258
- is_contaminant, 121, 129, 129, 130, 165, 185, 191, 192, 252
- is_decoy, 18, 121, 129, 130, 134, 165, 185, 191, 192, 252
- level_specific_contrasts, 21, 116, 127, 130, 163
- lfq_power_t_test_proteins, 45, 129, 153, 155, 205–208, 248
- lfq_power_t_test_quantiles_V2, 45, 129, 154, 154, 205–208, 248
- LFQData, 7, 9, 11, 12, 14, 15, 18, 24, 28, 29, 31, 32, 34–36, 51, 124, 125, 131, 142, 145, 148, 149
- LFQDataPlotter, 11, 12, 14, 15, 18, 137, 137, 145, 148, 149
- LFQDataStats, 11, 12, 14, 15, 18, 136, 137, 139, 142, 142, 148, 149
- LFQDataSummariser, 11, 12, 14, 15, 18, 137, 142, 145, 146, 149
- LFQDataToSummarizedExperiment, 11, 12, 14, 15, 18, 137, 142, 145, 148, 149
- LFQDataTransformer, 150
- linfct_from_model, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243,

- [245, 246, 251, 258](#)
- [linfct_matrix_contrasts, 23, 25, 27–30, 32–34, 36–38, 41–44, 48, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 156, 156, 158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- [list_facades, 23–25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 113, 117–119, 122, 125, 127, 128, 156, 157, 157, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- [list_to_AnalysisConfiguration, 158](#)
- [list_to_ContrastConfiguration, 20, 40, 47, 128, 159, 163, 164, 210, 213, 216, 253, 254](#)
- [lm, 234, 244](#)
- [lmer, 235](#)
- [lmFit, 31–33, 42, 242](#)
- [logistf, 236](#)
- [lookup_facade, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 113, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 211, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- [LR_test, 23, 25, 26, 28–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 201, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258](#)
- [main_effect_contrasts, 21, 116, 127, 131, 162](#)
- [make_interaction_column, 20, 40, 47, 128, 159, 163, 164, 210, 213, 216, 253, 254](#)
- [make_reduced_hierarchy_config, 20, 40, 47, 128, 159, 163, 164, 210, 213, 216, 253, 254](#)
- [matrix_to_tibble, 121, 129, 130, 164, 185, 191, 192, 252](#)
- [merge_contrasts_results, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 165, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- [missigness_histogram, 94, 118, 128, 166, 169, 170, 194–196, 198–200, 202–205, 256, 257](#)
- [MissingHelpers, 32, 167](#)
- [missingness_per_condition, 94, 118, 128, 167, 169, 170, 194–196, 198–200, 202–205, 256, 257](#)
- [missingness_per_condition_cumsum, 94, 118, 128, 167, 169, 169, 194–196, 198–200, 202–205, 256, 257](#)
- [Model, 23, 25–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 124, 125, 127, 128, 155, 157, 158, 160, 161, 166, 170, 174, 176, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258](#)
- [model_analyse, 23, 25–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178, 179, 180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- [model_summary, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107,](#)

- 111, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178, 179, 180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- ModelFirth, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 172, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- ModelInterface, 175
- ModelLimma, 23, 25, 27–31, 33–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 67, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 176, 179, 180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 240, 242–246, 251, 258
- moderated_p_deqms, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 181, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- moderated_p_deqms_long, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182, 182, 184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- moderated_p_limma, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 86, 89, 94, 96, 98, 100, 103, 105, 107, 111, 117–119, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182, 183, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- multigroup_volcano, 121, 129, 130, 165, 184, 191, 192, 252
- new_imputed_model, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 186, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- nr_B_in_A, 24, 114, 120, 129, 187, 213, 214
- nr_children_experiment, 188
- nr_features_experiment, 189
- nr_obs_sample, 190
- PACKAGE_DATA, 109, 110, 191, 209, 260
- pairs, 191
- pairs_smooth, 121, 129, 130, 165, 185, 191, 192, 252
- panel_cor, 121, 129, 130, 165, 185, 191, 192, 252
- pivot_model_contrasts_to_wide, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 192, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- plot_estimate, 16, 94, 112, 128, 167, 169, 170, 194, 195, 196, 198–200, 202–205, 256, 257
- plot_heatmap, 94, 128, 167, 169, 170, 194, 194, 196, 198–200, 202–205, 256, 257
- plot_heatmap_cor, 94, 128, 142, 167, 169, 170, 194, 195, 196, 198–200, 202–205, 256, 257
- plot_hierarchies_boxplot, 197
- plot_hierarchies_boxplot_df, 94, 128, 167, 169, 170, 194–196, 198, 199, 200, 202–205, 256, 257

- `plot_hierarchies_line`, 16, 94, 112, 128, 167, 169, 170, 194–196, 198, 199, 200, 202–205, 256, 257
- `plot_intensity_distribution_density` (`plot_intensity_distribution_violin`), 200
- `plot_intensity_distribution_violin`, 94, 128, 167, 169, 170, 194–196, 198, 199, 200, 202–205, 256, 257
- `plot_lrt_diagnostics`, 201
- `plot_na_heatmap`, 94, 128, 167, 169, 170, 194–196, 198–200, 201, 203–205, 256, 257
- `plot_pca`, 94, 128, 142, 167, 169, 170, 194–196, 198–200, 202, 204, 205, 256, 257
- `plot_raster`, 94, 128, 167, 169, 170, 194–196, 198–200, 202, 203, 205, 256, 257
- `plot_sample_correlation`, 94, 128, 167, 169, 170, 194–196, 198–200, 202–204, 204, 256, 257
- `plot_stat_density`, 45, 129, 154, 155, 205, 206–208, 248
- `plot_stat_density_median`, 45, 129, 154, 155, 205, 206, 207, 208, 248
- `plot_stat_violin`, 45, 129, 154, 155, 205, 206, 206, 208, 248
- `plot_stat_violin_median`, 45, 129, 154, 155, 205–207, 207, 208, 248
- `plot_stdv_vs_mean`, 45, 129, 154, 155, 205–208, 208, 248
- `poolvar` (`compute_pooled`), 44
- `prolfqua` (`prolfqua-package`), 6
- `prolfqua-package`, 6
- `prolfqua::AggregatorBase`, 10, 12–14
- `prolfqua::ContrastsFacadeBase`, 51, 52, 54, 60, 62, 69, 71, 73, 74, 76, 78, 80, 82, 83, 96, 97, 99, 104
- `prolfqua::ContrastsInterface`, 48, 51, 52, 54, 56, 58, 60, 62, 66, 69, 71, 73, 74, 76, 78, 80, 82, 83, 85, 88, 96, 97, 99, 101, 104, 106
- `prolfqua::ModelInterface`, 170, 172, 176
- `prolfqua::StrategyBase`, 229, 233, 235, 237, 238, 240
- `prolfqua_data`, 109, 110, 191, 209, 260
- `R6_extract_values`, 20, 40, 47, 128, 159, 163, 164, 209, 213, 216, 253, 254
- `rank_peptide_by_intensity`, 15, 210
- `register_facade`, 23–25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 113, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- `response_matrix_as_tibble`, 212
- `rfit`, 95, 97, 238
- `rlm`, 111, 218, 239
- `robust_scale`, 24, 114, 120, 129, 188, 213, 214
- `sample_subset`, 20, 40, 47, 128, 159, 163, 164, 210, 213, 216, 253, 254
- `scale_with_subset`, 24, 114, 120, 129, 188, 213, 214
- `script_copy_helper_vec`, 215
- `separate_hierarchy`, 20, 40, 47, 128, 159, 163, 164, 210, 213, 215, 216, 253, 254
- `setup_analysis`, 20, 40, 47, 128, 132, 159, 163, 164, 210, 213, 216, 216, 253, 254
- `sigma.rfit_prolfqua`, 23, 25, 27–30, 32–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 217, 219–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- `sigma.rlm`, 218
- `sim_build_models_lm`, 23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218, 219, 220, 221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258
- `sim_build_models_lmer`, 23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180,

- [182–184, 187, 193, 212, 218, 219, 220, 221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- `sim_build_models_logistf`, [23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231–234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- `sim_lfq_data`, [222](#)
- `sim_lfq_data_2factor_config`, [223](#)
- `sim_lfq_data_peptide_config`, [224](#)
- `sim_lfq_data_protein_config`, [224](#)
- `sim_make_model_lm`, [23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 225, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- `sim_make_model_lmer`, [23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 226, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- `strategy_binomial`, [52, 241](#)
- `strategy_limma`, [23, 25, 27–29, 31–38, 41–44, 50, 53–55, 57, 59, 61, 63, 68–75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 241, 243, 245, 246, 251, 258](#)
- `strategy_limpa`, [23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74–79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231–234, 236, 238–240, 242, 243, 245, 246, 251, 258](#)
- `strategy_lm`, [26, 52, 53, 81–84, 104, 230](#)
- `strategy_lm(strategy_logistf)`, [244](#)
- `strategy_lmer`, [26, 80, 171](#)
- `strategy_lmer(strategy_logistf)`, [244](#)
- `strategy_logistf`, [23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 244, 251, 258](#)
- `strategy_rfit`, [95–98](#)
- `strategy_rfit(strategy_logistf)`, [244](#)
- `strategy_rlm`, [99](#)
- `strategy_rlm(strategy_logistf)`, [244](#)
- `StrategyBase`, [227](#)
- `StrategyBinomial`, [227, 229, 241](#)
- `StrategyLimma`, [23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 230, 233, 234, 236, 237, 239–246, 251, 258](#)
- `StrategyLimpa`, [23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 232, 234, 236, 237, 239, 240, 242–246, 251, 258](#)
- `StrategyLM`, [23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128,](#)

- 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 233, 236–240, 242–246, 251, 258
- StrategyLmer, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 235, 237, 239, 240, 242–246, 251, 258
- StrategyLogistf, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 236, 239, 240, 242–246, 251, 258
- StrategyRfit, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 238, 240, 242–246, 251, 258
- StrategyRLM, 23, 25, 27–31, 33, 34, 36–38, 41–44, 49, 53, 55, 57, 59, 61, 62, 68, 70, 72, 73, 75, 77, 79, 81, 82, 84, 86, 89, 94, 96, 98, 100, 102, 105, 107, 111, 117–119, 122, 125, 127, 128, 155, 157, 158, 160, 161, 166, 171, 174, 178–180, 182–184, 186, 193, 211, 218–221, 226, 227, 231, 233, 234, 236, 237, 239, 239, 242–246, 251, 258
- summarize_hierarchy, 123, 124, 129, 247
- summarize_stats, 45, 129, 154, 155, 205–208, 248
- summarize_stats_factors, 249
- summarize_stats_quantiles (summarize_stats), 248
- summary_ROPECA_median_p_scaled, 23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 102, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 257
- table_facade, 121, 129, 130, 165, 185, 191, 192, 252
- table_factors, 20, 40, 47, 128, 159, 163, 164, 210, 213, 216, 253, 254
- table_factors_size, 20, 40, 47, 128, 159, 163, 164, 210, 213, 216, 253, 253
- tidy_to_wide_config, 254
- transform_work_intensity, 255
- upset_interaction_missing_stats, 94, 118, 128, 167, 169, 170, 194–196, 198–200, 202–205, 256, 257
- upset_missing_stats, 94, 118, 128, 167, 169, 170, 194–196, 198–200, 202–205, 256, 257
- vcov.rfit_prolfqua, 23, 25, 27–29, 31–34, 36–38, 41–44, 50, 53, 55, 57, 59, 61, 63, 68, 70, 72, 74, 75, 77, 79, 81, 83, 84, 87, 89, 94, 96, 99, 100, 103, 105, 107, 111, 117, 118, 120, 122, 125, 127, 128, 156–158, 160, 162, 166, 172, 174, 178–180, 182–184, 187, 193, 212, 218–221, 226, 227, 231, 233, 234, 236, 238–240, 242, 243, 245, 246, 251, 257
- volcano_plotly, 258
- which_missing, 259
- x5463yzwer453bbb, 109, 110, 191, 209, 260