

# Package: prolfqua (via r-universe)

October 9, 2026

**Type** Package

**Title** Proteomics Label Free Quantification

**Version** 0.99.2

**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 (>= 4.5)

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

**Config/pak/sysreqs**

cmake make libicu-dev libpng-dev libuv1-dev libssl-dev perl libx11-dev zlib1g-dev

**Repository** <https://biocstaging.r-universe.dev>

**Date/Publication** 2026-10-09 05:32:35 UTC

**RemoteUrl** <https://github.com/BiocStaging/prolfqua>

**RemoteRef** HEAD

**RemoteSha** e4e36d5bc553824ea82a5100ad6889b4e6d4f07d

## Contents

|                                          |    |
|------------------------------------------|----|
| adjust_p_values . . . . .                | 4  |
| AggregateLimpa . . . . .                 | 5  |
| AggregateMedpolish . . . . .             | 8  |
| AggregateRlm . . . . .                   | 9  |
| AggregateTopN . . . . .                  | 10 |
| AnalysisConfiguration . . . . .          | 12 |
| annotation_add_contrasts . . . . .       | 15 |
| AnovaExtractor . . . . .                 | 16 |
| build_contrast_analysis . . . . .        | 17 |
| build_model . . . . .                    | 19 |
| build_model_glm_peptide . . . . .        | 21 |
| build_model_glm_protein . . . . .        | 22 |
| build_model_impute . . . . .             | 23 |
| build_model_limma . . . . .              | 25 |
| build_model_limma_impute . . . . .       | 26 |
| build_model_limma_voom . . . . .         | 27 |
| build_model_limma_voom_impute . . . . .  | 29 |
| build_model_limpa . . . . .              | 30 |
| center_to_reference_cfg . . . . .        | 32 |
| ContrastConfiguration . . . . .          | 33 |
| Contrasts . . . . .                      | 35 |
| ContrastsBinomialNestedFacade . . . . .  | 38 |
| ContrastsDEqMSFacade . . . . .           | 40 |
| ContrastsDEqMSVoomFacade . . . . .       | 42 |
| ContrastsFirth . . . . .                 | 44 |
| ContrastsFirthFacade . . . . .           | 46 |
| ContrastsFirthNestedFacade . . . . .     | 48 |
| ContrastsInterface . . . . .             | 49 |
| ContrastsLimma . . . . .                 | 53 |
| ContrastsLimmaFacade . . . . .           | 56 |
| ContrastsLimmaImputeFacade . . . . .     | 58 |
| ContrastsLimmaVoomFacade . . . . .       | 60 |
| ContrastsLimmaVoomImputeFacade . . . . . | 62 |
| ContrastsLimpaFacade . . . . .           | 64 |

|                                            |     |
|--------------------------------------------|-----|
| ContrastsLimpaNestedFacade . . . . .       | 66  |
| ContrastsLmerNestedFacade . . . . .        | 68  |
| ContrastsLMFacade . . . . .                | 70  |
| ContrastsLMImputeFacade . . . . .          | 71  |
| ContrastsModerated . . . . .               | 73  |
| ContrastsModeratedDEqMS . . . . .          | 76  |
| ContrastsPlotter . . . . .                 | 79  |
| ContrastsRfitFacade . . . . .              | 85  |
| ContrastsRfitImputeFacade . . . . .        | 87  |
| ContrastsRLMFacade . . . . .               | 89  |
| ContrastsROPECA . . . . .                  | 90  |
| ContrastsROPECANestedFacade . . . . .      | 94  |
| ContrastsTable . . . . .                   | 96  |
| df.residual.rfit_prolfqua . . . . .        | 99  |
| FACADE_REGISTRY . . . . .                  | 100 |
| find_package_file . . . . .                | 101 |
| generate_contrasts . . . . .               | 101 |
| generate_contrasts_for_factor . . . . .    | 102 |
| get_uniprot_id_from_fasta_header . . . . . | 103 |
| group_label . . . . .                      | 104 |
| HierarchyCountsSample . . . . .            | 105 |
| impute_from_model . . . . .                | 106 |
| interaction_contrasts . . . . .            | 107 |
| INTERNAL_FUNCTIONS_BY_FAMILY . . . . .     | 108 |
| is_contaminant . . . . .                   | 109 |
| is_decoy . . . . .                         | 110 |
| level_specific_contrasts . . . . .         | 111 |
| LFQData . . . . .                          | 111 |
| LFQDataPlotter . . . . .                   | 118 |
| LFQDataStats . . . . .                     | 124 |
| LFQDataSummariser . . . . .                | 128 |
| LFQDataToSummarizedExperiment . . . . .    | 131 |
| LFQDataTransformer . . . . .               | 132 |
| list_facades . . . . .                     | 135 |
| list_to_AnalysisConfiguration . . . . .    | 136 |
| list_to_ContrastConfiguration . . . . .    | 137 |
| lookup_facade . . . . .                    | 138 |
| LR_test . . . . .                          | 139 |
| main_effect_contrasts . . . . .            | 141 |
| matrix_to_tibble . . . . .                 | 142 |
| merge_contrasts_results . . . . .          | 142 |
| MissingHelpers . . . . .                   | 144 |
| Model . . . . .                            | 145 |
| ModelFirth . . . . .                       | 148 |
| ModelInterface . . . . .                   | 150 |
| ModelLimma . . . . .                       | 152 |
| nr_children_experiment . . . . .           | 154 |
| nr_features_experiment . . . . .           | 155 |

|                                           |     |
|-------------------------------------------|-----|
| nr_obs_sample . . . . .                   | 156 |
| PACKAGE_DATA . . . . .                    | 157 |
| pivot_model_contrasts_to_wide . . . . .   | 158 |
| prolfqua_data . . . . .                   | 159 |
| R6_extract_values . . . . .               | 160 |
| register_facade . . . . .                 | 160 |
| setup_analysis . . . . .                  | 162 |
| sigma.rfit_prolfqua . . . . .             | 163 |
| sim_lfq_data . . . . .                    | 164 |
| sim_lfq_data_2factor_config . . . . .     | 165 |
| sim_lfq_data_peptide_config . . . . .     | 166 |
| sim_lfq_data_protein_config . . . . .     | 167 |
| strategy_binomial . . . . .               | 168 |
| strategy_limma . . . . .                  | 169 |
| strategy_limpa . . . . .                  | 170 |
| strategy_logistf . . . . .                | 171 |
| StrategyBinomial . . . . .                | 175 |
| StrategyLimma . . . . .                   | 176 |
| StrategyLimpa . . . . .                   | 178 |
| StrategyLM . . . . .                      | 180 |
| StrategyLmer . . . . .                    | 181 |
| StrategyLogistf . . . . .                 | 183 |
| StrategyRfit . . . . .                    | 185 |
| StrategyRLM . . . . .                     | 186 |
| summarize_stats_factors . . . . .         | 188 |
| table_facade . . . . .                    | 189 |
| upset_interaction_missing_stats . . . . . | 189 |
| vcov.rfit_prolfqua . . . . .              | 190 |
| volcano_plotly . . . . .                  | 191 |
| which_missing . . . . .                   | 193 |

## Index 194

---

|                 |                       |
|-----------------|-----------------------|
| 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**

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

---

AggregateLimpa

---

**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::AggregatBase -> 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](#), [AggregateRlm](#), [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())
```

---

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\\_configuration\(\)](#), [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](#), [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\\_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**

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

---

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**

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lfqdata</code>   | an <a href="#">LFQData</a> object                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>modelstr</code>  | model formula string without the response variable (e.g. <code>"~ group_"</code> ). The response is taken automatically from <code>lfqdata\$get_config()\$get_response()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>contrasts</code> | named character vector of contrasts (e.g. <code>c("A_vs_B" = "group_A - group_B")</code> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>method</code>    | 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 <a href="#">register_facade</a> . The authoritative list is <code>names(list_facades())</code> . Defaults to <code>"lm"</code> . |
| <code>...</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\\_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\\_value\(\)](#), [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`, `StrategyLimpafacade`, `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_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_s`, `vcov.rfit_prolfqua()`

**Examples**

```

D <- prolfqua::sim_lfq_data_peptide_config(Nprot = 20, weight_missing = 0.1)
D$data$abundance |> is.na() |> sum()
D <- prolfqua::sim_lfq_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 <a href="#">LFQData</a> object                                               |
| modelstr | model formula string without the response variable (e.g. " <code>~ group_</code> ") |

## Value

a [ModelFirth](#) 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](#), [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_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_s,
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 <a href="#">LFQData</a> 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](#), [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_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_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_s,
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 <a href="#">compute_borrowed_variance</a> via <a href="#">impute_refit_singular</a> . "fall-back" (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](#), [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\\_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\\_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\\_value\(\)](#), [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 <a href="#">LFQData</a> object        |
| strategy   | output of <a href="#">strategy_limma</a> |
| 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](#), [StrategyLimpafacade](#), [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\\_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\\_value\(\)](#), [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 <a href="#">LFQData</a> object (aggregated to protein level)                                                                                                                                                                 |
| strategy   | output of <a href="#">strategy_limma</a>                                                                                                                                                                                        |
| model_name | name of model (default: strategy name + "Imputed")                                                                                                                                                                              |
| lod        | numeric limit of detection; if NULL, auto-computed from data via <a href="#">MissingHelpers</a>                                                                                                                                 |
| df_method  | how to set degrees of freedom for imputed proteins: "observed" (default) uses $\max(n_{\text{observed}} - p, 1)$ where $n_{\text{observed}}$ 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](#), [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\\_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\\_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\\_value\(\)](#), [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 <a href="#">LFQData</a> object (aggregated to protein level) |
| strategy   | output of <a href="#">strategy_limma</a>                        |
| 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/TODO_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](#), [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\\_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_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 <- 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 <a href="#">LFQData</a> object (aggregated to protein level)                                                                                                       |
| strategy   | output of <a href="#">strategy_limma</a>                                                                                                                              |
| 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\\_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\\_s\(\)](#), [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

*Build limpa vooma model from aggregated LFQData*

---

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

```

---

center\_to\_reference\_cfg

*center to reference*

---

### Description

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

### Usage

```
center_to_reference_cfg(lfqdata, lfqdareference)
```

### Arguments

lfqdata            LFQData object containing the data to center  
lfqdareference    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()

```

## 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\\_col\(\)](#), [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 | <i>Estimate contrasts using Wald Test</i> |
|-----------|-------------------------------------------|

---

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](#), [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\\_lim](#), [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))

```

**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](#), [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\\_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)
```

```
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\\_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)
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())
```

---

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\\_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

```
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()
p1 <- 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](#), [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\\_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 = 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](#), [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\\_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  $\times$  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 sign(effect) \* -log10(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 ( $|effect| > diff\_threshold$ ). For backends whose [ContrastConfiguration](#) has significance\_directional = TRUE (e.g. SAINTexpress, where only positive enrichment is biologically meaningful), the filter is one-sided (effect > 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 | <i>Limma-based contrasts (direct limma pipeline)</i> |
|----------------|------------------------------------------------------|

---

## 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 = p.adjust,
  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](#), [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\\_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_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**

```
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`, `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_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

```

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\\_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

```
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_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`, `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\\_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**

```
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\\_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\\_value\(\)](#), [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`, `ContrastsLimmaImputeFacade`, `ContrastsLimmaVoomFacade`, `ContrastsLimmaVoomImputeFacade`, `ContrastsLimpafacade`, `ContrastsLimpafacadeNestedFacade`, `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_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_s`, `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\\_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()
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()
```

---

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\\_lim](#), [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      *Limma moderated contrasts*

---

## 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](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpaFacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [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\\_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\\_value\(\)](#), [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()

```

---

ContrastsModeratedDEqMS

*DEqMS count-dependent moderated contrasts*

---

## 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](#), [ContrastsLimpaNestedFacade](#), [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_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_s,
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()

```

**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

fcthresh 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", fcthresh = 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](#), [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_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_s,
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](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [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\\_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\\_value\(\)](#), [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\_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](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [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\\_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\\_value\(\)](#), [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\\_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\\_s\(\)](#), [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

*ROPECA reproducibility-optimization method*


---

**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](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [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\\_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(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.based.significance` to `FDR`.

Columns not directly produced by ROPECA are derived heuristically:

- `std.error` = `diff / statistic` (algebraic:  $t = \text{estimate} / \text{SE}$ )
- `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](#), [ContrastsLimpaNestedFacade](#), [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_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_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_s,
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\\_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_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)
```

---

df.residual.rfit\_prolfqua

*Residual degrees of freedom for an augmented rfit fit*


---

## 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](#), [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\\_limma\(\)](#), [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\\_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\\_value\(\)](#), [vcov.rfit\\_prolfqua\(\)](#)

## Examples

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

---

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
```

---

|                   |                                    |
|-------------------|------------------------------------|
| 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_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 | <i>group label function</i> |
|-------------|-----------------------------|

---

**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](#), [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\\_limma\(\)](#), [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\\_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\\_value\(\)](#), [vcov\\_rfit\\_prolfqua\(\)](#)

**Examples**

group\_label("a", "b")

---

HierarchyCountsSample *Count distinct elements for each level of hierarchy per sample*

---

**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())
```

---

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| 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

|                      |                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <code>model</code>   | a <a href="#">Model</a> with <code>lm</code> fits, as returned by <a href="#">build_model</a> or <a href="#">build_model_impute</a> |
| <code>lfqdata</code> | the <a href="#">LFQData</a> the model was fitted on (same response and <code>subject_id</code> )                                    |

### 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\\_limma\(\)](#), [compute\\_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\\_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\\_s\(\)](#), [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))
```

---

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\\_column\(\)](#), [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, ContrastsLimpaNestedFacade, ContrastsLmerNestedFacade, ContrastsModerated, ContrastsModeratedDEqMS, ContrastsPlotter, ContrastsRLMFacade, ContrastsROPECA, ContrastsROPECANestedFacade, ContrastsRfitFacade, ContrastsRfitImputeFacade, ContrastsTable, 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\_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\_s, 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, upset\_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

Detect contaminant identifiers

---

## 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 than 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$config)
lfqdata$is_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 | <i>Summarize LFQData</i> |
|-------------------|--------------------------|

---

## 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

*Decorate LFQData with Methods for transforming Intensities***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()

```

**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\\_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\(\)](#), [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**

```
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](#), [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_lim,
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(),
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_limp(), 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](#), [ContrastsLimmaImputeFacade](#), [ContrastsLimmaVoomFacade](#), [ContrastsLimmaVoomImputeFacade](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL\\_FUNCTIONS\\_BY\\_FAMILY](#), [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\\_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\\_s\(\)](#), [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)
```

---

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| 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 | <i>Merge contrast results coming from two different model.</i> |
|-------------------------|----------------------------------------------------------------|

---

### 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\\_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\(\)](#), [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()

```

---

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()
```

---

Model

*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 -> 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\\_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\\_value\(\)](#), [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 = prolfqua::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](#), [ContrastsLimpafacade](#), [ContrastsLimpaNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL\\_FUNCTIONS\\_BY\\_FAMILY](#), [LR\\_test\(\)](#), [Model](#), [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\\_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\\_value\(\)](#), [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 `limfct` 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](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL\\_FUNCTIONS\\_BY\\_FAMILY](#), [LR\\_test\(\)](#), [Model](#), [ModelFirth](#), [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\\_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\\_s\(\)](#), [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()
```

---

nr\_children\_experiment

*Max nr\_children across samples per hierarchy unit*

---

**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\\_checksummarizationrobust87](#), [data\\_checksummarizerobust69](#), [data\\_ionstar](#), [prolfqua\\_data\(\)](#), [x5463yzwer453bbb](#)

---

`pivot_model_contrasts_to_wide`

*pivot model contrasts matrix to wide format produced by ‘contrasts\_linfct’ and ...*

---

## 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

|                                          |                                                       |
|------------------------------------------|-------------------------------------------------------|
| <code>model_interaction_contrasts</code> | data.frame with contrast results in long format       |
| <code>subject_id</code>                  | column name(s) identifying subjects (e.g. protein_Id) |
| <code>columns</code>                     | character vector of value columns to pivot wide       |
| <code>contrast</code>                    | 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\\_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(), 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

```
# 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)
```

---

|               |                                |
|---------------|--------------------------------|
| 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\\_co](#), [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)
```

---

|                 |                                         |
|-----------------|-----------------------------------------|
| 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 e.g. protein -> protein FC, peptide -> peptide FC) or "nested" (facade takes child-level input and emits parent-level contrasts, e.g. 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 <a href="#">lookup_facade</a> .                                                                                                                                   |

**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](#), [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\\_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\(\)](#), [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)
```

---

|                |                                                                            |
|----------------|----------------------------------------------------------------------------|
| setup_analysis | Setup a tidy table compatible with a <a href="#">AnalysisConfiguration</a> |
|----------------|----------------------------------------------------------------------------|

---

## 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](#), [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\\_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\(\)](#), [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**

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

---

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))
```

---

|                   |                                                 |
|-------------------|-------------------------------------------------|
| 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 <a href="#">eBayes</a>                                                                                                                                          |
| robust     | logical, passed to <a href="#">eBayes</a>                                                                                                                                          |
| weights    | either a character string (column name in annotation for per-sample weights) or a numeric matrix (proteins x samples) passed to <a href="#">lmFit</a> . 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](#), [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_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_limma("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 <a href="#">eBayes</a>           |
| robust     | logical, passed to <a href="#">eBayes</a>           |
| 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\\_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\\_logistf\(\)](#), [summary\\_ROPECA\\_median\\_p\\_scaled\(\)](#), [vcov.rfit\\_prolfqua\(\)](#)

**Examples**

```
strat <- strategy_limpafacade("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 <a href="#">lm</a> . 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](#), [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\\_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, 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\_lim, 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\_limp(), 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, ContrastsLimpFacade, ContrastsLimpNestedFacade, 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\_lim, 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\_limp(), 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, ContrastsLimpFacade, ContrastsLimpNestedFacade, 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\_lim

```
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\_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(), 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

```

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](#), [ContrastsLimpafacadeNestedFacade](#), [ContrastsLmerNestedFacade](#), [ContrastsModerated](#), [ContrastsModeratedDEqMS](#), [ContrastsPlotter](#), [ContrastsRLMFacade](#), [ContrastsROPECA](#), [ContrastsROPECANestedFacade](#), [ContrastsRfitFacade](#), [ContrastsRfitImputeFacade](#), [ContrastsTable](#), [INTERNAL\\_FUNCTIONS\\_BY\\_FAMILY](#), [LR\\_test\(\)](#), [Model](#), [ModelFirth](#), [ModelLimma](#), [StrategyLM](#), [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_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,
vcov.rfit_prolfqua()
```

Examples

```
strat <- StrategyLimma$new("abundance ~ group_")
strat$formula
strat$model_name
```

---

|               |                                              |
|---------------|----------------------------------------------|
| StrategyLimpa | <i>R6 class for limpa modelling strategy</i> |
|---------------|----------------------------------------------|

---

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](#), [ContrastsLimpaNestedFacade](#), [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\\_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\\_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\\_value\(\)](#), [vcov.rfit\\_prolfqua\(\)](#)

**Examples**

```
strat <- StrategyLimp$new("abundance ~ group_")
strat$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](#), [ContrastsLimpafacadeNestedFacade](#), [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\\_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\\_s](#), [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](#), [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](#), [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_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_s,
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

*Firth's logistic regression strategy (R6 class)*


---

## 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](#), [ContrastsLimpaNestedFacade](#), [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\(\)](#), [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_s
vcov.rfit_prolfqua()
```

## Examples

```
strat <- StrategyLogistf$new("bin_resp ~ condition")
strat$formula
```

---

|              |                                                  |
|--------------|--------------------------------------------------|
| StrategyRfit | <i>Rank-based regression strategy (R6 class)</i> |
|--------------|--------------------------------------------------|

---

## 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](#), [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](#), [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\\_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\\_value\(\)](#), [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](#), [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](#), [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.rfitprolfqua\(\)](#), [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_s,
vcov.rfit_prolfqua()
```

## Examples

```
strat <- StrategyRLM$new("Intensity ~ condition", model_name = "parallel design")
strat$formula
```

---

```
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))
```

---

|              |                                                      |
|--------------|------------------------------------------------------|
| 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")
```

---

|                                 |                                                   |
|---------------------------------|---------------------------------------------------|
| upset_interaction_missing_stats | <i>UpSetR plot from interaction_missing_stats</i> |
|---------------------------------|---------------------------------------------------|

---

**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)
```

---

|                    |                                                                   |
|--------------------|-------------------------------------------------------------------|
| vcov.rfit_prolfqua | <i>Named variance-covariance matrix for an augmented rfit fit</i> |
|--------------------|-------------------------------------------------------------------|

---

**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](#), [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\\_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\(\)](#)

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

|                              |                                                                                                                                                                                            |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.data</code>           | data frame                                                                                                                                                                                 |
| <code>effect</code>          | effect size x-axis                                                                                                                                                                         |
| <code>significance</code>    | significance                                                                                                                                                                               |
| <code>contrast</code>        | column with contrast labels                                                                                                                                                                |
| <code>proteinID</code>       | column with protein ids                                                                                                                                                                    |
| <code>color</code>           | column used for colouring points                                                                                                                                                           |
| <code>palette</code>         | named colour vector for the colour aesthetic                                                                                                                                               |
| <code>xintercept</code>      | vertical abline at x                                                                                                                                                                       |
| <code>yintercept</code>      | horizontal abline at y                                                                                                                                                                     |
| <code>minsignificance</code> | 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. |
| <code>title_size</code>      | font size of the subplot title annotation                                                                                                                                                  |
| <code>group</code>           | 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))
```

# Index

- \* **LFQData**
  - AggregateLimpa, [5](#)
  - AggregateMedpolish, [8](#)
  - AggregateRlm, [9](#)
  - AggregateTopN, [10](#)
  - LFQData, [111](#)
  - LFQDataPlotter, [118](#)
  - LFQDataStats, [124](#)
  - LFQDataSummariser, [128](#)
  - LFQDataToSummarizedExperiment, [131](#)
- \* **aggregation**
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, [108](#)
- \* **configuration**
  - AnalysisConfiguration, [12](#)
  - ContrastConfiguration, [33](#)
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, [108](#)
  - list\_to\_ContrastConfiguration, [137](#)
  - R6\_extract\_values, [160](#)
  - setup\_analysis, [162](#)
- \* **contrasts**
  - annotation\_add\_contrasts, [15](#)
  - generate\_contrasts, [101](#)
  - generate\_contrasts\_for\_factor, [102](#)
  - interaction\_contrasts, [107](#)
  - level\_specific\_contrasts, [111](#)
  - main\_effect\_contrasts, [141](#)
- \* **datasets**
  - FACADE\_REGISTRY, [100](#)
- \* **data**
  - PACKAGE\_DATA, [157](#)
  - prolfqua\_data, [159](#)
- \* **imputation**
  - upset\_interaction\_missing\_stats, [189](#)
- \* **modelling**
  - AnovaExtractor, [16](#)
  - build\_contrast\_analysis, [17](#)
  - build\_model, [19](#)
  - build\_model\_glm\_peptide, [21](#)
  - build\_model\_glm\_protein, [22](#)
  - build\_model\_impute, [23](#)
  - build\_model\_limma, [25](#)
  - build\_model\_limma\_impute, [26](#)
  - build\_model\_limma\_voom, [27](#)
  - build\_model\_limma\_voom\_impute, [29](#)
  - build\_model\_limpa, [30](#)
  - Contrasts, [35](#)
  - ContrastsDEqMSFacade, [40](#)
  - ContrastsDEqMSVoomFacade, [42](#)
  - ContrastsFirth, [44](#)
  - ContrastsFirthFacade, [46](#)
  - ContrastsFirthNestedFacade, [48](#)
  - ContrastsLimma, [53](#)
  - ContrastsLimmaFacade, [56](#)
  - ContrastsLimmaImputeFacade, [58](#)
  - ContrastsLimmaVoomFacade, [60](#)
  - ContrastsLimmaVoomImputeFacade, [62](#)
  - ContrastsLimpaFacade, [64](#)
  - ContrastsLimpaNestedFacade, [66](#)
  - ContrastsLmerNestedFacade, [68](#)
  - ContrastsLMFacade, [70](#)
  - ContrastsLMImputeFacade, [71](#)
  - ContrastsModerated, [73](#)
  - ContrastsModeratedDEqMS, [76](#)
  - ContrastsPlotter, [79](#)
  - ContrastsRfitFacade, [85](#)
  - ContrastsRfitImputeFacade, [87](#)
  - ContrastsRLMFacade, [89](#)
  - ContrastsROPECA, [90](#)
  - ContrastsROPECANestedFacade, [94](#)
  - ContrastsTable, [96](#)
  - df.residual.rfit\_prolfqua, [99](#)
  - group\_label, [104](#)
  - impute\_from\_model, [106](#)
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, [108](#)
  - list\_facades, [135](#)
  - lookup\_facade, [138](#)
  - LR\_test, [139](#)

- merge\_contrasts\_results, 142
- Model, 145
- ModelFirth, 148
- ModelLimma, 152
- pivot\_model\_contrasts\_to\_wide, 158
- register\_facade, 160
- sigma.rfit\_prolfqua, 163
- strategy\_limma, 169
- strategy\_limpa, 170
- strategy\_logistf, 171
- StrategyLimma, 176
- StrategyLimpa, 178
- StrategyLM, 180
- StrategyLmer, 181
- StrategyLogistf, 183
- StrategyRfit, 185
- StrategyRLM, 186
- vcov.rfit\_prolfqua, 190
- \* plotting**
  - ContrastsPlotter, 79
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, 108
  - upset\_interaction\_missing\_stats, 189
- \* preprocessing**
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, 108
- \* stats**
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, 108
- \* summary**
  - HierarchyCountsSample, 105
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, 108
- \* utilities**
  - get\_uniprot\_id\_from\_fasta\_header, 103
  - INTERNAL\_FUNCTIONS\_BY\_FAMILY, 108
  - is\_contaminant, 109
  - is\_decoy, 110
  - matrix\_to\_tibble, 142
  - table\_facade, 189
- adjust\_p\_values, 4
- aggregate\_intensity\_top\_n, 108
- AggregateLimpa, 5, 8, 10, 11, 30, 31, 64–67, 117, 120, 123, 126, 130, 131
- AggregateMedpolish, 7, 8, 10, 11, 117, 123, 126, 130, 131
- AggregateRlm, 7, 8, 9, 11, 117, 123, 126, 130, 131
- AggregateTopN, 7, 8, 10, 10, 117, 123, 126, 130, 131
- AggregatorBase, 7, 8, 10, 11, 117, 123, 126, 130, 131
- AnalysisConfiguration, 12, 33, 35, 108, 138, 160, 162
- annotation\_add\_contrasts, 15, 102, 108, 111, 141
- AnovaExtractor, 16, 18, 20–22, 24, 25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 95, 98, 99, 104, 107, 108, 136, 138, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- apply\_to\_response\_matrix, 109
- build\_contrast\_analysis, 17, 17, 20, 21, 23–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- build\_model, 17, 18, 19, 21, 23–25, 27, 28, 30, 31, 36, 37, 40, 41, 43–45, 47, 49, 55, 57, 59, 61, 63, 65, 67–73, 75, 78, 83, 85, 86, 88–90, 93, 94, 96, 98, 99, 104, 106, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- build\_model\_glm\_peptide, 17, 18, 20, 21, 23–25, 27, 28, 30, 31, 37, 41, 43, 45, 47–49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- build\_model\_glm\_protein, 17, 18, 20, 21, 22, 24, 25, 27, 28, 30, 31, 37, 41, 43, 45–47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- build\_model\_impute, 17, 18, 20, 21, 23, 23,

- 25–28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 84, 86–88, 90, 93, 96, 98, 99,  
104, 106, 107, 109, 136, 139, 140,  
143, 144, 147, 149, 154, 158, 161,  
164, 169, 171–174, 177, 179, 181,  
182, 184, 186, 187, 191
- `build_model_limma`, 17, 18, 20, 21, 23, 24,  
25, 27, 28, 30, 31, 37, 41, 43, 45, 47,  
49, 55–57, 59, 61, 63, 65, 67, 69, 71,  
73, 75, 78, 84, 86, 88, 90, 93, 96, 98,  
99, 104, 107, 109, 136, 139, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 176, 177, 179, 181,  
182, 184, 186, 187, 191
- `build_model_limma_impute`, 17, 18, 20, 21,  
23–25, 26, 28–31, 37, 41, 43, 45, 47,  
49, 55, 57–59, 61, 63, 65, 67, 69, 71,  
73, 75, 78, 84, 86, 88, 90, 93, 96, 98,  
99, 104, 107, 109, 136, 139, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- `build_model_limma_voom`, 17, 18, 20, 21,  
23–25, 27, 27, 30, 31, 37, 41–43, 45,  
47, 49, 55, 57, 59–61, 63, 65, 67, 69,  
71, 73, 75, 78, 84, 86, 88, 90, 93, 96,  
98, 99, 104, 107, 109, 136, 139, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- `build_model_limma_voom_impute`, 17, 18,  
20, 21, 23–25, 27, 28, 29, 31, 37, 41,  
43, 45, 47, 49, 55, 57, 59, 61–63, 65,  
67, 69, 71, 73, 75, 78, 84, 86, 88, 90,  
93, 96, 98, 99, 104, 107, 109, 136,  
139, 140, 143, 147, 149, 154, 158,  
161, 164, 169, 171–174, 177, 179,  
181, 182, 184, 186, 187, 191
- `build_model_limpa`, 17, 18, 20, 21, 23–25,  
27, 28, 30, 30, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63–67, 69, 71, 73, 75,  
78, 84, 86, 88, 90, 93, 96, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177–179, 181, 182, 184,  
186, 187, 191
- `build_model_logistf`, 17, 18, 20, 21, 23–25,  
27, 28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 84, 86, 88, 90, 93, 96, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- `center_to_reference_cfg`, 32
- `complete_cases`, 14, 35, 108, 138, 160, 162
- `compute_borrowed_variance`, 17, 18, 20,  
22–25, 27, 28, 30, 31, 37, 41, 43, 45,  
47, 49, 55, 57, 59, 61, 63, 65, 67, 69,  
71, 73, 75, 78, 84, 86, 88, 90, 93, 96,  
98, 99, 104, 107, 109, 136, 139, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 183,  
184, 186, 187, 191
- `compute_borrowed_variance_limma`, 17, 18,  
20, 22–25, 27, 28, 30, 31, 37, 41, 43,  
45, 47, 49, 55, 57, 59, 61, 63, 65, 67,  
69, 71, 73, 75, 78, 84, 86, 88, 90, 93,  
96, 98, 99, 104, 107, 109, 136, 139,  
140, 143, 147, 149, 154, 158, 161,  
164, 169, 171–174, 177, 179, 181,  
183, 184, 186, 187, 191
- `compute_contrast`, 17, 18, 20, 22–25, 27, 28,  
30, 31, 35, 37, 41, 43, 45, 47, 49, 55,  
57, 59, 61, 63, 65, 67, 69, 71, 73, 75,  
78, 84, 86, 88, 90, 93, 96, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 183, 184,  
186, 187, 191
- `compute_lmer_contrast`, 17, 18, 20, 22–25,  
27, 28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 84, 86, 88, 90, 93, 96, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 178, 179, 181–184, 186,  
187, 191
- `compute_pooled`, 109
- `ContrastConfiguration`, 14, 33, 50, 51, 108,  
137, 138, 160, 162
- `Contrasts`, 17, 18, 20–22, 24, 25, 27, 28, 30,  
31, 33, 35, 40, 41, 43, 45, 47, 49, 53,  
55, 57, 59, 61, 63, 65, 67–73, 75, 78,  
83, 85–90, 93–95, 98, 99, 104, 107,

- 108, 136, 138, 140, 142, 143, 147,  
149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- ContrastsBinomialNestedFacade, 18, 38
- ContrastsDEqMSFacade, 17, 18, 20–22, 24,  
25, 27, 28, 30, 31, 37, 40, 43, 45, 47,  
49, 55, 57, 59, 61, 63, 65, 67, 69, 71,  
73, 75, 78, 83, 86, 88, 90, 93, 95, 98,  
99, 104, 107, 108, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsDEqMSVoomFacade, 17, 18, 20–22,  
24, 25, 27, 28, 30, 31, 37, 41, 42, 45,  
47, 49, 55, 57, 59, 61, 63, 65, 67, 69,  
71, 73, 75, 78, 83, 86, 88, 90, 93, 95,  
98, 99, 104, 107, 108, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsFacadeBase, 17, 18, 20–22, 24, 25,  
27, 28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 83, 86, 88, 90, 93, 95, 98, 99,  
104, 107, 108, 136, 138, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- ContrastsFirth, 17, 18, 20–22, 24, 25, 27,  
28, 30, 31, 37, 41, 43, 44, 46–49, 55,  
57, 59, 61, 63, 65, 67, 69, 71, 73, 75,  
78, 83, 86, 88, 90, 93, 95, 98, 99,  
104, 107, 108, 136, 138, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- ContrastsFirthFacade, 17, 18, 20–22, 24,  
25, 27, 28, 30, 31, 37, 41, 43, 45, 46,  
48, 49, 55, 57, 59, 61, 63, 65, 67, 69,  
71, 73, 75, 78, 83, 86, 88, 90, 93, 95,  
98, 99, 104, 107, 108, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsFirthNestedFacade, 17, 18,  
20–22, 24, 25, 27, 28, 30, 31, 37, 39,  
41, 43, 45–47, 48, 55, 57, 59, 61, 63,  
65, 67, 69, 71, 73, 75, 78, 83, 86, 88,  
90, 93, 95, 98, 99, 104, 107, 108,  
136, 138, 140, 143, 147, 149, 154,  
158, 161, 164, 169, 171–174, 177,  
179, 181, 182, 184, 186, 187, 191
- ContrastsInterface, 49, 53
- ContrastsLimma, 17, 18, 20–22, 24, 25, 27,  
28, 30, 31, 33, 37, 41–43, 45, 47, 49,  
53, 56–67, 69, 71, 73, 75, 78, 83, 86,  
88, 90, 93, 95, 98, 99, 104, 107, 108,  
136, 138, 140, 143, 147, 149, 154,  
158, 161, 164, 169, 171–174, 177,  
179, 181, 182, 184, 186, 187, 191
- ContrastsLimmaFacade, 17, 18, 20–22, 24,  
25, 27, 28, 30, 31, 37, 41, 43, 45, 47,  
49, 55, 56, 59, 61, 63, 65, 67, 69, 71,  
73, 75, 78, 83, 86, 88, 90, 93, 95, 98,  
99, 104, 107, 108, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsLimmaImputeFacade, 17, 18,  
20–22, 24, 25, 27, 28, 30, 31, 37, 41,  
43, 45, 47, 49, 55, 57, 58, 61, 63, 65,  
67, 69, 71, 73, 75, 78, 83, 86, 88, 90,  
93, 95, 98, 99, 104, 107, 108, 136,  
138, 140, 143, 147, 149, 154, 158,  
161, 164, 169, 171–174, 177, 179,  
181, 182, 184, 186, 187, 191
- ContrastsLimmaVoomFacade, 17, 18, 20–22,  
24, 25, 27, 28, 30, 31, 37, 41, 43, 45,  
47, 49, 55, 57, 59, 60, 63, 65, 67, 69,  
71, 73, 75, 78, 83, 86, 88, 90, 93, 95,  
98, 99, 104, 107, 109, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsLimmaVoomImputeFacade, 17, 18,  
20–22, 24, 25, 27, 28, 30, 31, 37, 41,  
43, 45, 47, 49, 55, 57, 59, 61, 62, 65,  
67, 69, 71, 73, 75, 78, 83, 86, 88, 90,  
93, 95, 98, 99, 104, 107, 109, 136,  
138, 140, 143, 147, 149, 154, 158,  
161, 164, 169, 171–174, 177, 179,  
181, 182, 184, 186, 187, 191
- ContrastsLimpaFacade, 17, 18, 20–22, 24,  
25, 27, 28, 30, 31, 37, 41, 43, 45, 47,  
49, 55, 57, 59, 61, 63, 64, 66, 67, 69,

- 71, 73, 75, 78, 83, 86, 88, 90, 93, 95,  
98, 99, 104, 107, 109, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsLimpaNestedFacade, 17, 18,  
20–22, 24, 25, 27, 28, 30, 31, 37, 41,  
43, 45, 47, 49, 55, 57, 59, 61, 63–65,  
66, 69, 71, 73, 75, 78, 83, 86, 88, 90,  
93, 95, 98, 99, 104, 107, 109, 136,  
138, 140, 143, 147, 149, 154, 158,  
161, 164, 169, 171–174, 177, 179,  
181, 182, 184, 186, 187, 191
- ContrastsLmerNestedFacade, 17, 18, 20–22,  
24, 25, 27, 28, 30, 31, 37, 41, 43, 45,  
47, 49, 55, 57, 59, 61, 63, 65, 67, 68,  
71, 73, 75, 78, 83, 86, 88, 90, 93, 95,  
98, 99, 104, 107, 109, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsLMFacade, 17, 18, 20–22, 24, 25,  
27, 28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 70, 73,  
75, 78, 83, 85, 86, 88, 90, 93, 95, 98,  
99, 104, 107, 108, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsLMImputeFacade, 17, 18, 20–22,  
24, 25, 27, 28, 30, 31, 37, 41, 43, 45,  
47, 49, 55, 57, 59, 61, 63, 65, 67, 69,  
71, 71, 75, 78, 83, 86–88, 90, 93, 95,  
98, 99, 104, 107, 108, 136, 138, 140,  
142, 143, 147, 149, 154, 158, 161,  
164, 169, 171–174, 177, 179, 181,  
182, 184, 186, 187, 191
- ContrastsModerated, 17, 18, 20–22, 24, 25,  
27, 28, 30, 31, 33, 37, 39, 41, 43, 45,  
47, 49, 55, 57, 59, 61, 63, 65, 67–72,  
73, 73, 77, 78, 83, 85–90, 93, 95, 98,  
99, 104, 107, 109, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsModeratedDEqMS, 17, 18, 20–22,  
24, 25, 27, 28, 30, 31, 37, 40–43, 45,  
47, 49, 53–55, 57, 59, 61, 63, 65, 67,  
69, 71, 73, 75, 76, 83, 86, 88, 90, 93,  
95, 98, 99, 104, 107, 109, 136, 138,  
140, 143, 147, 149, 154, 158, 161,  
164, 169, 171–174, 177, 179, 181,  
182, 184, 186, 187, 191
- ContrastsPlotter, 17, 18, 20–22, 24, 25, 27,  
28, 30, 31, 37, 41, 43, 45, 47, 49–51,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 79, 86, 88, 90, 92, 93, 95,  
97–99, 104, 107, 109, 136, 138, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 190, 191
- ContrastsRfitFacade, 17, 18, 20–22, 24, 25,  
27, 28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 83, 85, 88, 90, 93, 95, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- ContrastsRfitImputeFacade, 17, 18, 20, 21,  
23–25, 27, 28, 30, 31, 37, 41, 43, 45,  
47, 49, 55, 57, 59, 61, 63, 65, 67, 69,  
71, 73, 75, 78, 83, 86, 87, 90, 93, 95,  
98, 99, 104, 107, 109, 136, 139, 140,  
143, 147, 149, 154, 158, 161, 164,  
169, 171–174, 177, 179, 181, 182,  
184, 186, 187, 191
- ContrastsRLMFacade, 17, 18, 20–22, 24, 25,  
27, 28, 30, 31, 37, 41, 43, 45, 47, 49,  
55, 57, 59, 61, 63, 65, 67, 69, 71, 73,  
75, 78, 83, 86, 88, 89, 93, 95, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- ContrastsROPECA, 17, 18, 20–22, 24, 25, 27,  
28, 30, 31, 37, 41, 43, 45, 47, 49, 55,  
57, 59, 61, 63, 65, 67, 69, 71, 73, 75,  
78, 83, 86, 88, 90, 90, 94, 95, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 187, 191
- ContrastsROPECANestedFacade, 17, 18,  
20–22, 24, 25, 27, 28, 30, 31, 37, 41,  
43, 45, 47, 49, 55, 57, 59, 61, 63, 65,

- 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 94, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- ContrastsTable, 17, 18, 20, 21, 23–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 95, 96, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191
- data\_checksuarizationrobust87, 157, 159
- data\_checksuarizerobust69, 157, 159
- data\_ionstar, 157, 159
- df.residual.rfit\_prolfqua, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 178, 179, 181, 183, 184, 186, 187, 191
- eBayes, 169, 170, 177, 178
- estimate\_intensity, 108
- FACADE\_REGISTRY, 100, 136
- facet\_wrap, 82
- filter\_proteins\_by\_peptide\_count, 109
- find\_package\_file, 101
- generate\_contrasts, 15, 101, 102, 108, 111, 141
- generate\_contrasts\_for\_factor, 15, 102, 102, 108, 111, 141
- get\_anova\_df, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 169, 171–174, 178, 179, 181, 183, 184, 186, 187, 191
- get\_complete\_model\_fit, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 187, 191
- get\_contrast, 190
- get\_p\_values\_pbeta, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 187, 191
- get\_robscales, 109
- get\_uniprot\_id\_from\_fasta\_header, 103, 109, 110, 142, 189
- glm, 175
- group\_label, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 187, 191
- hierarchy\_counts, 106, 109
- HierarchyCountsSample, 105, 109
- impute\_from\_model, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 106, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 187, 191
- impute\_refit\_singular, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 187, 191
- interaction\_contrasts, 15, 102, 107, 111, 141

- INTERNAL\_FUNCTIONS\_BY\_FAMILY, [14](#), [17](#), [18](#),  
[20](#), [21](#), [23–25](#), [27](#), [28](#), [30](#), [31](#), [35](#), [37](#),  
[41](#), [43](#), [45](#), [47](#), [49](#), [55](#), [57](#), [59](#), [61](#), [63](#),  
[65](#), [67](#), [69](#), [71](#), [73](#), [75](#), [78](#), [83](#), [84](#), [86](#),  
[88](#), [90](#), [93](#), [95](#), [98](#), [99](#), [103](#), [104](#), [106](#),  
[107](#), [108](#), [110](#), [136](#), [138–140](#), [142](#),  
[143](#), [147](#), [149](#), [154](#), [158](#), [160–162](#),  
[164](#), [169](#), [171–174](#), [177](#), [179](#), [181](#),  
[182](#), [184](#), [186](#), [187](#), [189–191](#)
- is\_contaminant, [103](#), [109](#), [109](#), [110](#), [142](#), [189](#)
- is\_decoy, [12](#), [103](#), [109](#), [110](#), [110](#), [114](#), [142](#),  
[189](#)
- level\_specific\_contrasts, [15](#), [102](#), [108](#),  
[111](#), [141](#)
- lfq\_power\_t\_test\_proteins, [109](#)
- lfq\_power\_t\_test\_quantiles\_V2, [109](#)
- LFQData, [7](#), [8](#), [10](#), [11](#), [18](#), [21](#), [22](#), [25](#), [26](#), [28](#),  
[29](#), [31](#), [39](#), [106](#), [107](#), [111](#), [123](#), [126](#),  
[130](#), [131](#)
- LFQDataPlotter, [7](#), [8](#), [10](#), [11](#), [117](#), [118](#), [126](#),  
[130](#), [131](#)
- LFQDataStats, [7](#), [8](#), [10](#), [11](#), [117](#), [120](#), [123](#),  
[124](#), [130](#), [131](#)
- LFQDataSummariser, [7](#), [8](#), [10](#), [11](#), [117](#), [123](#),  
[126](#), [128](#), [131](#)
- LFQDataToSummarizedExperiment, [7](#), [8](#), [10](#),  
[11](#), [117](#), [123](#), [126](#), [130](#), [131](#)
- LFQDataTransformer, [132](#)
- linfct\_from\_model, [17](#), [18](#), [20](#), [22–25](#), [27](#),  
[28](#), [30](#), [31](#), [37](#), [41](#), [43](#), [46](#), [47](#), [49](#), [55](#),  
[57](#), [59](#), [61](#), [63](#), [65](#), [67](#), [69](#), [71](#), [73](#), [75](#),  
[79](#), [84](#), [86](#), [88](#), [90](#), [93](#), [96](#), [98](#), [99](#),  
[104](#), [107](#), [109](#), [136](#), [139](#), [140](#), [143](#),  
[147](#), [149](#), [154](#), [159](#), [161](#), [164](#),  
[170–174](#), [178](#), [179](#), [181](#), [183](#), [184](#),  
[186](#), [187](#), [191](#)
- linfct\_matrix\_contrasts, [17](#), [18](#), [20](#),  
[22–25](#), [27](#), [28](#), [30](#), [31](#), [35](#), [37](#), [41](#), [43](#),  
[46](#), [47](#), [49](#), [55](#), [57](#), [59](#), [61](#), [63](#), [65](#), [67](#),  
[69](#), [71](#), [73](#), [75](#), [79](#), [84](#), [86](#), [88](#), [90](#), [93](#),  
[96](#), [98](#), [99](#), [104](#), [107](#), [109](#), [136](#), [139](#),  
[140](#), [143](#), [147](#), [149](#), [154](#), [159](#), [161](#),  
[164](#), [170–174](#), [178](#), [179](#), [181](#), [183](#),  
[184](#), [186](#), [187](#), [191](#)
- list\_facades, [17](#), [18](#), [20](#), [22–25](#), [27](#), [28](#), [30](#),  
[31](#), [37](#), [41](#), [43](#), [46](#), [47](#), [49](#), [55](#), [57](#), [59](#),  
[61](#), [63](#), [65](#), [67](#), [69](#), [71](#), [73](#), [75](#), [79](#), [84](#),  
[86](#), [88](#), [90](#), [93](#), [96](#), [98–100](#), [104](#), [107](#),  
[109](#), [135](#), [139](#), [140](#), [143](#), [147](#), [149](#),  
[154](#), [159](#), [161](#), [164](#), [170–174](#), [178](#),  
[179](#), [181](#), [183](#), [184](#), [186](#), [187](#), [191](#)
- list\_to\_AnalysisConfiguration, [136](#)
- list\_to\_ContrastConfiguration, [14](#), [35](#),  
[108](#), [137](#), [160](#), [162](#)
- lm, [172](#), [180](#)
- lmer, [181](#)
- lmFit, [25–27](#), [169](#)
- logistf, [183](#)
- lookup\_facade, [17](#), [18](#), [20](#), [22–25](#), [27](#), [28](#), [30](#),  
[31](#), [37](#), [41](#), [43](#), [46](#), [47](#), [49](#), [55](#), [57](#), [59](#),  
[61](#), [63](#), [65](#), [67](#), [69](#), [71](#), [73](#), [75](#), [79](#), [84](#),  
[86](#), [88](#), [90](#), [93](#), [96](#), [98–100](#), [104](#), [107](#),  
[109](#), [136](#), [138](#), [140](#), [143](#), [147](#), [149](#),  
[154](#), [159–161](#), [164](#), [170–174](#), [178](#),  
[179](#), [181](#), [183](#), [184](#), [186](#), [187](#), [191](#)
- LR\_test, [17](#), [18](#), [20](#), [21](#), [23–25](#), [27](#), [28](#), [30](#), [31](#),  
[37](#), [41](#), [43](#), [45](#), [47](#), [49](#), [55](#), [57](#), [59](#), [61](#),  
[63](#), [65](#), [67](#), [69](#), [71](#), [73](#), [75](#), [78](#), [83](#), [86](#),  
[88](#), [90](#), [93](#), [95](#), [98](#), [99](#), [104](#), [107](#), [109](#),  
[136](#), [139](#), [139](#), [143](#), [147](#), [149](#), [154](#),  
[158](#), [161](#), [164](#), [169](#), [171–174](#), [177](#),  
[179](#), [181](#), [182](#), [184](#), [186](#), [187](#), [191](#)
- main\_effect\_contrasts, [15](#), [102](#), [108](#), [111](#),  
[141](#)
- make\_interaction\_column, [14](#), [35](#), [108](#), [138](#),  
[160](#), [162](#)
- make\_reduced\_hierarchy\_config, [14](#), [35](#),  
[108](#), [138](#), [160](#), [162](#)
- matrix\_to\_tibble, [103](#), [109](#), [110](#), [142](#), [189](#)
- merge\_contrasts\_results, [17](#), [18](#), [20](#),  
[22–25](#), [27](#), [28](#), [30](#), [31](#), [37](#), [41](#), [43](#), [46](#),  
[47](#), [49](#), [55](#), [57](#), [59](#), [61](#), [63](#), [65](#), [67](#), [69](#),  
[71](#), [73](#), [75](#), [79](#), [84](#), [86](#), [88](#), [90](#), [93](#), [96](#),  
[98](#), [99](#), [104](#), [107](#), [109](#), [136](#), [139](#), [140](#),  
[142](#), [147](#), [149](#), [154](#), [159](#), [161](#), [164](#),  
[170–174](#), [178](#), [179](#), [181](#), [183](#), [184](#),  
[186](#), [187](#), [191](#)
- missigness\_histogram, [84](#), [109](#), [190](#)
- MissingHelpers, [26](#), [144](#)
- missingness\_per\_condition, [84](#), [109](#), [190](#)
- missingness\_per\_condition\_cumsum, [84](#),  
[109](#), [190](#)
- Model, [17](#), [18](#), [20](#), [21](#), [23–25](#), [27](#), [28](#), [30](#), [31](#),  
[37](#), [41](#), [43](#), [45](#), [47](#), [49](#), [55](#), [57](#), [59](#), [61](#),  
[63](#), [65](#), [67](#), [69](#), [71](#), [73](#), [75](#), [78](#), [83](#), [86](#),  
[88](#), [90](#), [93](#), [95](#), [98](#), [99](#), [104](#), [106](#), [107](#),

- 109, 136, 139, 140, 143, 145, 149, 152, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191*
- model\_analyse, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 188, 191*
- model\_summary, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 188, 191*
- ModelFirth, 17, 18, 20–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 95, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 148, 154, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191*
- ModelInterface, 150*
- Modellimma, 17, 18, 20, 21, 23–25, 27, 28, 30, 31, 37, 41, 43, 45, 47, 49, 54, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 83, 86, 88, 90, 93, 95, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 152, 158, 161, 164, 169, 171–174, 177, 179, 181, 182, 184, 186, 187, 191*
- moderated\_p\_deqms, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 188, 191*
- moderated\_p\_deqms\_long, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 78, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 188, 191*
- moderated\_p\_limma, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 184, 186, 188, 191*
- multigroup\_volcano, 103, 109, 110, 142, 189*
- new\_imputed\_model, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191*
- nr\_B\_in\_A, 109*
- nr\_children\_experiment, 154*
- nr\_features\_experiment, 155*
- nr\_obs\_sample, 156*
- PACKAGE\_DATA, 157, 159*
- pairs\_smooth, 103, 109, 110, 142, 189*
- panel\_cor, 103, 109, 110, 142, 189*
- pivot\_model\_contrasts\_to\_wide, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 158, 161, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191*
- plot\_estimate, 84, 108, 109, 190*
- plot\_heatmap, 84, 109, 190*
- plot\_heatmap\_cor, 84, 109, 123, 190*
- plot\_hierarchies\_boxplot\_df, 84, 109, 190*
- plot\_hierarchies\_line, 84, 108, 109, 190*
- plot\_intensity\_distribution\_violin, 84, 109, 190*
- plot\_na\_heatmap, 84, 109, 190*
- plot\_pca, 84, 109, 123, 190*
- plot\_raster, 84, 109, 190*
- plot\_sample\_correlation, 84, 109, 190*

- `plot_stat_density`, 109
- `plot_stat_density_median`, 109
- `plot_stat_violin`, 109
- `plot_stat_violin_median`, 109
- `plot_stdv_vs_mean`, 109
- `prolfqua::AggregatorBase`, 6, 8–10
- `prolfqua::ContrastsFacadeBase`, 39, 40, 42, 46, 48, 56, 58, 60, 62, 64, 66, 68, 70, 72, 85, 87, 89, 94
- `prolfqua::ContrastsInterface`, 35, 39, 40, 42, 44, 46, 48, 53, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 77, 85, 87, 89, 91, 94, 96
- `prolfqua::ModelInterface`, 146, 148, 152
- `prolfqua::StrategyBase`, 175, 180, 182, 183, 185, 187
- `prolfqua_data`, 157, 159
- `R6_extract_values`, 14, 35, 108, 137, 138, 160, 162
- `register_facade`, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98–100, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 160, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `rfit`, 85, 87, 185
- `rlm`, 186
- `robust_scale`, 109
- `sample_subset`, 14, 35, 108, 138, 160, 162
- `scale_with_subset`, 109
- `separate_hierarchy`, 14, 35, 108, 138, 160, 162
- `setup_analysis`, 14, 35, 108, 113, 138, 160, 162
- `sigma.rfit_prolfqua`, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 163, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `sim_build_models_lm`, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `sim_build_models_lmer`, 17, 18, 20, 22–25, 27, 28, 30, 31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `sim_build_models_logistf`, 17, 18, 20, 22–25, 27, 29–31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 161, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `sim_lfq_data`, 164
- `sim_lfq_data_2factor_config`, 165
- `sim_lfq_data_peptide_config`, 166
- `sim_lfq_data_protein_config`, 167
- `sim_make_model_lm`, 17, 18, 20, 22–25, 27, 29–31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 162, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `sim_make_model_lmer`, 17, 18, 20, 22–25, 27, 29–31, 37, 41, 43, 46, 47, 49, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 162, 164, 170–174, 178, 179, 181, 183, 185, 186, 188, 191
- `strategy_binomial`, 39, 168
- `strategy_limma`, 17, 18, 20, 22–31, 37, 41–43, 46, 47, 49, 55–63, 65, 67, 69, 71, 73, 75, 79, 84, 86, 88, 90, 93, 96, 98, 99, 104, 107, 109, 136, 139, 140, 143, 147, 149, 154, 159, 162, 164, 169, 171–174, 177–179, 181, 183,

- 185, 186, 188, 191
- strategy\_limpa, 17, 18, 20, 22–25, 27,  
29–31, 37, 41, 43, 46, 47, 49, 55, 57,  
59, 61, 63–67, 69, 71, 73, 75, 79, 84,  
86, 88, 90, 93, 96, 98, 99, 104, 107,  
109, 136, 139, 140, 143, 147, 149,  
154, 159, 162, 164, 170, 170,  
172–174, 178, 179, 181, 183, 185,  
186, 188, 191
- strategy\_lm, 20, 40, 41, 70, 72, 94, 95, 176
- strategy\_lm(strategy\_logistf), 171
- strategy\_lmer, 20, 68, 69, 146
- strategy\_lmer(strategy\_logistf), 171
- strategy\_logistf, 17, 18, 20, 22–25, 27,  
29–31, 37, 41, 43, 46, 47, 49, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 79,  
84, 86, 88, 90, 93, 96, 98, 99, 104,  
107, 109, 136, 139, 140, 143, 147,  
149, 154, 159, 162, 164, 170, 171,  
171, 178, 179, 181, 183, 185, 186,  
188, 191
- strategy\_rfit, 85–88
- strategy\_rfit(strategy\_logistf), 171
- strategy\_rlm, 89
- strategy\_rlm(strategy\_logistf), 171
- StrategyBinomial, 168, 175
- StrategyLimma, 17, 18, 20, 21, 23–25, 27, 28,  
30, 31, 37, 41, 43, 45, 47, 49, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 78,  
83, 86, 88, 90, 93, 95, 98, 99, 104,  
107, 109, 136, 139, 140, 143, 147,  
149, 154, 158, 161, 164, 169,  
171–174, 176, 179, 181, 182, 184,  
186, 187, 191
- StrategyLimpa, 17, 18, 20, 21, 23–25, 27, 28,  
30, 31, 37, 41, 43, 45, 47, 49, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 78,  
83, 86, 88, 90, 93, 95, 98, 99, 104,  
107, 109, 136, 139, 140, 143, 147,  
149, 154, 158, 161, 164, 169–174,  
177, 178, 181, 182, 184, 186, 187,  
191
- StrategyLM, 17, 18, 20, 21, 23–25, 27, 28, 30,  
31, 37, 41, 43, 45, 47, 49, 55, 57, 59,  
61, 63, 65, 67, 69, 71, 73, 75, 78, 83,  
86, 88, 90, 93, 95, 98, 99, 104, 107,  
109, 136, 139, 140, 143, 147, 149,  
154, 158, 161, 164, 169, 171–174,  
177, 179, 180, 182, 184–187, 191
- StrategyLmer, 17, 18, 20, 21, 23–25, 27, 28,  
30, 31, 37, 41, 43, 45, 47, 49, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 78,  
83, 86, 88, 90, 93, 95, 98, 99, 104,  
107, 109, 136, 139, 140, 143, 147,  
149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 181, 184,  
186, 187, 191
- StrategyLogistf, 17, 18, 20, 21, 23–25, 27,  
28, 30, 31, 37, 41, 43, 45, 47, 49, 55,  
57, 59, 61, 63, 65, 67, 69, 71, 73, 75,  
78, 83, 86, 88, 90, 93, 95, 98, 99,  
104, 107, 109, 136, 139, 140, 143,  
147, 149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 183,  
186, 187, 191
- StrategyRfit, 17, 18, 20, 21, 23–25, 27, 28,  
30, 31, 37, 41, 43, 45, 47, 49, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 78,  
83, 86, 88, 90, 93, 96, 98, 99, 104,  
107, 109, 136, 139, 140, 143, 147,  
149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
185, 187, 191
- StrategyRLM, 17, 18, 20, 21, 23–25, 27, 28,  
30, 31, 37, 41, 43, 45, 47, 49, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 78,  
83, 86, 88, 90, 93, 95, 98, 99, 104,  
107, 109, 136, 139, 140, 143, 147,  
149, 154, 158, 161, 164, 169,  
171–174, 177, 179, 181, 182, 184,  
186, 186, 191
- summarize\_hierarchy, 106, 109
- summarize\_stats, 109
- summarize\_stats\_factors, 188
- summary\_ROPECA\_median\_p\_scaled, 17, 18,  
20, 22–25, 27, 29–31, 37, 41, 43, 46,  
47, 49, 55, 57, 59, 61, 63, 65, 67, 69,  
71, 73, 75, 79, 84, 86, 88, 90, 93, 96,  
98, 99, 104, 107, 109, 136, 139, 140,  
143, 147, 149, 154, 159, 162, 164,  
170–174, 178, 179, 181, 183, 185,  
186, 188, 191
- table\_facade, 103, 109, 110, 142, 189
- table\_factors, 14, 35, 108, 138, 160, 162
- table\_factors\_size, 14, 35, 108, 138, 160,  
162

upset\_interaction\_missing\_stats, [84](#),  
[109](#), [189](#)  
upset\_missing\_stats, [84](#), [109](#), [190](#)  
  
vcov.rfit\_prolfqua, [17](#), [18](#), [20](#), [22–25](#), [27](#),  
[29–31](#), [37](#), [41](#), [43](#), [46](#), [47](#), [49](#), [55](#), [57](#),  
[59](#), [61](#), [63](#), [65](#), [67](#), [69](#), [71](#), [73](#), [75](#), [79](#),  
[84](#), [86](#), [88](#), [90](#), [93](#), [96](#), [98](#), [99](#), [104](#),  
[107](#), [109](#), [136](#), [139](#), [140](#), [143](#), [147](#),  
[149](#), [154](#), [159](#), [162](#), [164](#), [170–174](#),  
[178](#), [179](#), [181](#), [183](#), [185](#), [186](#), [188](#),  
[190](#)  
volcano\_plotly, [191](#)  
  
which\_missing, [193](#)  
  
x5463yzwer453bbb, [157](#), [159](#)