Package: looOmics (via r-universe)

November 16, 2026

Type Package

Title Leave-One-Out Sensitivity Analysis for Omics Differential Models

Version 0.99.0

Description Performs sample-wise leave-one-out sensitivity analysis for differential models on high-dimensional omics data. Quantifies changes in feature-level coefficients, standard errors, statistical significance, and within-term feature rankings after individual samples are omitted; flags influential samples; and evaluates associations between influence and biological or technical sample metadata. Supports linear and generalized linear models, limma, and DESeq2, with optional Bioconductor data adapters and future-based parallel execution.

License MIT + file LICENSE

Encoding UTF-8

Collate 'generics.R' 'class-OmicsInput.R' 'class-LOOResult.R' 'accessors.R' 'adapters.R' 'engines.R' 'flag.R' 'influence.R' 'loo-core.R' 'plot.R' 'profile.R'

Depends R (>= 4.3.0)

Imports methods, stats, SummarizedExperiment, withr, future (>= 1.33.0), future.apply (>= 1.11.0), ggplot2, broom, digest, tibble

Suggests testthat (>= 3.0.0), knitr, rmarkdown, Biobase, phyloseq, limma, DESeq2, BiocStyle, S4Vectors

Config/testthat/edition 3

VignetteBuilder knitr

biocViews DifferentialExpression, RNASeq, Microbiome, QualityControl, StatisticalMethod

Config/roxygen2/version 8.1.0

URL https://github.com/btanay041/looOmics

BugReports https://github.com/btanay041/looOmics/issues

Config/pak/sysreqs libicu-dev zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-11-16 02:00:07 UTC

RemoteUrl https://github.com/BiocStaging/looOmics

RemoteRef HEAD

RemoteSha 129105b3f5a07f3a1125287ecf734e695f4925e6

Contents

looOmics-package . . . 2
as_omics_input . . . 3
assay,OmicsInput,missing-method . . . 4
autoplot.LOOResult . . . 5
coef,LOOResult-method . . . 5
flag_influential . . . 6
influence_by_term . . . 7
loo_sensitivity . . . 8
LOOResult-class . . . 9
LOOResult-methods . . . 10
OmicsInput-class . . . 11
print.FlagProfile . . . 12
profile_flags . . . 12
profiles . . . 14
reexports . . . 15
register_engine . . . 15
run_engine . . . 16
tidy.LOOResult . . . 17

Index **18**

`looOmics-package` *Leave-One-Out Sensitivity Analysis for Omics Differential Models*

Description

looOmics performs sample-wise leave-one-out (LOO) revalidation of differential-analysis pipelines on gene expression and microbiome data. For each sample it refits the model on the remaining n-1 samples, records coefficient shifts for every feature and model term, computes per-sample influence metrics, flags influential samples, and profiles flagged samples against biological and technical metadata covariates.

The analysis is a sensitivity analysis: an influential sample may reflect genuine biological heterogeneity and is not automatically a "bad" sample.

Details

The main entry point is `loo_sensitivity`. Inputs are built with `as_omics_input`, which dispatches on the input class (matrix, data.frame, and - when installed - SummarizedExperiment, ExpressionSet, phyloseq). Results are stored in a `LOOResult` object; influence summaries per model term are available through `influence_by_term`.

Workflow:

1. `res <- loo_sensitivity(se, ~ condition + batch)`
2. `res <- flag_influential(res)`
3. `res <- profile_flags(res)`
4. `plot(res)`

`as_omics_input` *Coerce an omics object into the unified input container*

Description

Dispatches on the class of `x`. Matrix and data.frame methods are always available; SummarizedExperiment, ExpressionSet and phyloseq methods require the corresponding packages. SummarizedExperiment is the recommended primary interface for Bioconductor workflows. The model formula is stored for formula-driven engines such as DESeq2. Assay values must be numeric and finite; missing and infinite values should be handled upstream.

Usage

```
as_omics_input(x, formula, engine = "auto", transform = "none",
               pseudocount = 1, ...)
```

Arguments

`x`	An omics object.
`formula`	Model formula evaluated in the sample covariate table; the left-hand side is implicit.
`engine`	Engine id (see `loo_sensitivity`). "auto" resolves deterministically to lm. Select limma or DESeq2 explicitly when appropriate for the assay type.
`transform`	Transformation applied to the assay: "none" (default), "log2", or "clr". The pseudocount is added exactly once, inside this transformation.
`pseudocount`	Pseudocount added before log2/clr transforms (default 1).
`...`	For the matrix method: `col_data` (sample-level covariates) and optionally `row_data`. For SummarizedExperiment: `assay_name`.

Value

An `OmicsInput` object.

Examples

```
mat <- matrix(1:12, 3, 4, dimnames = list(paste0("f", 1:3), paste0("S", 1:4)))
cd <- data.frame(grp = factor(c("A","A","B","B")), row.names = colnames(mat))
inp <- as_omics_input(mat, ~ grp, col_data = cd)
```

`assay,OmicsInput,missing-method`

Extract the assay matrix from an OmicsInput

Description

Extract the assay matrix from an OmicsInput

Usage

```
## S4 method for signature 'OmicsInput,missing'
assay(x, i, withDimnames = TRUE, ...)
```

Arguments

`x`	An `OmicsInput`.
`...`	Unused.

Value

The numeric assay matrix (features in rows and samples in columns).

Examples

```
mat <- matrix(1:12, 3, 4,
              dimnames = list(paste0("f", 1:3), paste0("S", 1:4)))
cd <- data.frame(group = factor(c("A", "A", "B", "B")),
                 row.names = colnames(mat))
input <- as_omics_input(mat, ~ group, col_data = cd)
assay(input)
```

autoplot.LOOResult *Plot a LOOResult*

Description

Visualizes per-sample influence scores, highlighting flagged samples.

Usage

```
## S3 method for class 'LOOResult'
autoplot(object, type = c("shift", "scree"), ...)
## S4 method for signature 'LOOResult,missing'
plot(x, y, ...)
```

Arguments

object, x	A LOOResult.
type	One of "shift" (per-sample score barplot, flagged in red) or "scree" (ranked influence scores).
y, ...	Unused / passed to layers.

Value

A ggplot object.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 6)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
res <- flag_influential(res, method = "topk", k = 2)
autoplot(res, type = "shift")
```

coef,LOOResult-method *Extract coefficients from a LOOResult*

Description

Extract coefficients from a LOOResult

Usage

```
## S4 method for signature 'LOOResult'
coef(object, ...)
```

Arguments

object	A LOOResult.
...	Unused.

Value

A data.frame of full-sample feature-level coefficients, standard errors, p-values, and model terms.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(8))
x <- matrix(rnorm(10 * 8), nrow = 10,
            dimnames = list(paste0("gene", 1:10), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 4)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
head(coef(res))
```

---

flag_influential *Flag influential samples*

Description

Applies a flagging rule to the per-sample influence scores. Flagged samples merit inspection; they are not automatically invalid.

Usage

```
flag_influential(object, method = c("mad", "threshold", "topk", "quantile"),
                 threshold = NULL, k = NULL, q = 0.9, metric = "max_dfbetas")
```

Arguments

object	A LOOResult with influence metrics.
method	One of "mad" (robust z on max |DFBETAS|), "threshold" (absolute cutoff, default 1), "topk", or "quantile".
threshold	Numeric; MAD cutoff (default 5) or absolute cutoff (default 1) depending on method.
k	Number of top samples for "topk".
q	Quantile for "quantile" (default 0.9).
metric	Aggregate to flag on: "max_dfbetas" (default), "n_sig_flips", or "median_abs_delta".

Value

The input object with the populated `flags` slot (columns: sample, score, metric, method, flagged).

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 6)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
res <- flag_influential(res, method = "topk", k = 2)
res@flags
```

`influence_by_term` *Per-term influence scores*

Description

Returns the per-sample influence aggregates computed separately for each model term. The overall `influence` slot pools all terms; this accessor shows which model terms drive a sample's score. Pass `terms` filtering downstream if the intercept is not of interest.

Usage

```
influence_by_term(object)
```

Arguments

`object`	A `LOOResult`.

Value

A data.frame with columns term, sample, max_abs_shift, n_sig_flips.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 6)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
head(influence_by_term(res))
```

`loo_sensitivity` *Sample-wise leave-one-out sensitivity analysis for omics models*

Description

For each sample i, refits the model on the remaining n-1 samples and records coefficient shifts for every feature and model term, then computes influence metrics per sample. The entire relevant pipeline (normalization and model-specific estimation steps) is rerun for every leave-one-out fit where appropriate.

Usage

```
loo_sensitivity(x, formula, engine = "auto", parallel = TRUE,
                seed = TRUE, ...)
```

Arguments

`x`	Omics object: matrix/data.frame (features x samples, with `col_data`), or an OmicsInput. SummarizedExperiment, ExpressionSet and phyloseq are supported when those packages are installed.
`formula`	Model formula evaluated in the sample covariate table; the left-hand side is implicit (each feature is the response).
`engine`	Engine id: "auto" (default; resolves deterministically to lm), "lm", "glm", "limma", "deseq2", or a name registered via `register_engine`.
`parallel`	Logical, default TRUE. If FALSE, runs sequentially via lapply. The parallel strategy itself is owned by the user via `future::plan()`.
`seed`	Integer or TRUE, passed to future.apply::future_lapply.
`...`	Input-adapter arguments (such as `col_data`, `assay_name`, `transform`, and `pseudocount`) and engine-specific arguments (such as `family` for glm). Adapter arguments are not forwarded to the model-fitting engine.

Details

The column `dfbetas` in the shift table is a deletion-based standardized coefficient shift, (beta_full - beta_loo) / se_loo. The shift table also reports full and leave-one-out p-values, BH-adjusted q-values, and within-term feature ranks. A positive rank_change means the feature moved to a numerically larger (weaker) rank after sample deletion. For ordinary linear models this coincides with classical DFBETAS; for other engines it is the analogous standardized deletion shift and should be interpreted as such. This is a sensitivity analysis for detecting influential samples, not an automatic bad-sample detector: an influential sample may reflect genuine biological heterogeneity.

Failed leave-one-out refits (e.g. rank-deficient designs after sample deletion) are recorded in the `failures` slot and do not abort the run.

Value

A `LOOResult` with full-sample coefficients, LOO betas/SEs, the long shift table, and per-sample influence scores.

Examples

```
set.seed(1)
mat <- matrix(rnorm(100 * 12), 100, 12,
              dimnames = list(paste0("g", 1:100), paste0("S", 1:12)))
cd <- data.frame(condition = factor(rep(c("A", "B"), 6)),
                 batch = factor(rep(1:2, 6)), row.names = colnames(mat))
res <- loo_sensitivity(mat, ~ condition, col_data = cd)
flag_influential(res)
```

`LOOResult-class` *Leave-one-out sensitivity analysis result*

Description

S4 container for the output of `loo_sensitivity`. Holds the full-sample fit, leave-one-out betas/SEs, the long shift table (per feature, term, and left-out sample: beta, beta_loo, delta, dfbetas, p_loo, q_loo, sig_flip), per-sample influence aggregates, failed-refit records, flags (`flag_influential`), metadata profiles (`profile_flags`), and diagnostics (runtime, future plan, RNG seed, sample metadata, per-term influence).

Value

An S4 object of class `LOOResult`, containing the full-sample model results, leave-one-out estimates, influence statistics, failed-refit records, flags, metadata profiles, and diagnostics.

Slots

`call` The matched call.

`engine` Engine id.

`engine_info` Engine metadata (package versions, terms).

`coefficients` Full-sample fit: feature, term, beta, se, p.

`loo_betas`, `loo_se` LOO fits; rows are feature::term, columns are left-out samples.

`shifts` Long shift table.

`influence` Per-sample aggregate scores.

`failures` Failed LOO refits (fail-soft record).

`flags` Output of flag_influential().

`profiles` Output of profile_flags().

`diagnostics` Runtime, seed, future plan, metadata, per-term influence.

`input_hash` Digest of the input assay and metadata.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(8))
x <- matrix(rnorm(10 * 8), nrow = 10,
            dimnames = list(paste0("gene", 1:10), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 4)),
                 row.names = samples)
result <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
is(result, "LOOResult")
```

LOOResult-methods *Methods for LOOResult objects*

Description

Standard methods for `LOOResult` objects.

Value

'show' returns invisibly; 'summary' returns a per-term data.frame; 'coef' returns the full-sample coefficient data.frame.

Methods

`signature(object = "LOOResult")` `coef` returns the full-sample coefficient table.

`show` prints a compact summary (engine, dimensions, flags, profiles).

`summary` prints per-term |DFBETAS| shift summaries.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(8))
x <- matrix(rnorm(10 * 8), nrow = 10,
            dimnames = list(paste0("gene", 1:10), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 4)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
show(res)
head(summary(res))
head(coef(res))
```

`OmicsInput-class` *Unified omics input container*

Description

S4 container holding the assay (features x samples), feature metadata, sample-level covariates, engine id, applied transformation, provenance class, and the deparsed model formula (used by formula-driven engines such as DESeq2). Constructed via `as_omics_input`.

Value

An S4 object of class `OmicsInput`, containing the assay matrix, feature metadata, sample metadata, engine, transformation, provenance information, and model formula.

Slots

`assay` Numeric matrix, features x samples.

`row_data` Feature metadata (data.frame).

`col_data` Sample-level covariates (data.frame).

`engine` Engine id (character).

`transform` Recorded transformation.

`data_class` Provenance class of the original object.

`formula` Deparsed model formula.

Examples

```
mat <- matrix(
  seq_len(12), nrow = 3,
  dimnames = list(paste0("feature", 1:3), paste0("sample", 1:4))
)
metadata <- data.frame(
  condition = factor(c("control", "control", "treated", "treated")),
  row.names = colnames(mat)
)
input <- as_omics_input(mat, ~ condition, col_data = metadata)
is(input, "OmicsInput")
```

print.FlagProfile *Print a biological and technical metadata profile*

Description

Prints univariate associations between influential-sample status and sample metadata, optional multivariable logistic-regression results, and reasons why selected covariates could not be tested.

Usage

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

Arguments

x	A FlagProfile object returned by profiles().
...	Additional arguments (currently unused).

Value

The input object, invisibly.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(
  condition = factor(rep(c("A", "B"), each = 6)),
  batch = factor(rep(c("batch1", "batch2"), 6)),
  row.names = samples
)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
res <- flag_influential(res, method = "topk", k = 3)
res <- profile_flags(res, multivariable = FALSE)
print(profiles(res))
```

profile_flags *Profile flagged samples against biological and technical covariates*

Description

Evaluates associations between influential-sample status and biological or technical sample metadata. Continuous covariates are assessed with Wilcoxon rank-sum tests; categorical covariates use chi-square tests or simulated Fisher tests when expected counts are sparse. Univariate p-values are adjusted using Benjamini-Hochberg. An optional multivariable logistic regression estimates adjusted associations across selected metadata covariates. Numeric covariates are standardized to one standard deviation in the adjusted model; factor coefficients are relative to their reference levels. Variables used in the differential model are retained and labeled because their association with influence can be biologically informative. This is exploratory association analysis, not causal inference. Covariates that cannot be tested are reported with reasons.

Usage

```
profile_flags(object, covariates = NULL, model_covariates = NULL,
              multivariable = TRUE, min_cell_size = 3,
              seed = 1L, fisher_B = 2000L)
```

Arguments

`object`	A `LOOResult` with flags computed.
`covariates`	Character vector of sample-metadata columns to profile. Defaults to all available metadata, including variables in the differential model.
`model_covariates`	Character vector of variables in the original differential model. Defaults to variables inferred from its formula. These variables remain profiled and are labeled as in-model.
`multivariable`	Logical; fit an exploratory adjusted logistic regression when the sample size and design support it.
`min_cell_size`	Minimum observations per categorical level before rare levels are collapsed into `Other`.
`seed`	Integer seed for simulated Fisher tests. The caller's random number generator state is restored after each simulated test.
`fisher_B`	Number of Monte Carlo replicates for simulated Fisher tests.

Value

The input object with the populated `profiles` slot; retrieve it with `profiles`.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(
  condition = factor(rep(c("A", "B"), each = 6)),
  batch = factor(rep(c("batch1", "batch2"), 6)),
  age = seq(40, 62, length.out = 12),
```

```
  row.names = samples
)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
res <- flag_influential(res, method = "topk", k = 3)
res <- profile_flags(res, covariates = c("batch", "age"),
                     multivariable = FALSE)
profiles(res)
```

`profiles` *Get the flag profile from a LOOResult*

Description

Returns the metadata-profiling results computed by `profile_flags`.

Usage

```
profiles(object)
```

Arguments

`object`	A `LOOResult`.

Value

The FlagProfile computed by `profile_flags` (NULL if not yet computed).

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(
  condition = factor(rep(c("A", "B"), each = 6)),
  batch = factor(rep(c("batch1", "batch2"), 6)),
  age = seq(40, 62, length.out = 12),
  row.names = samples
)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
res <- flag_influential(res, method = "topk", k = 3)
res <- profile_flags(res, covariates = c("batch", "age"),
                     multivariable = FALSE)
profiles(res)
```

reexports *Objects re-exported from other packages*

Description

These objects are imported from other packages and re-exported by looOmics so that `autoplot` and `tidy` can be used directly on `LOOResult` objects. See `autoplot.LOOResult` and `tidy.LOOResult` for the looOmics methods.

Value

'autoplot()' returns a ggplot object; 'tidy()' returns a tibble.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 6)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
autoplot(res)
head(tidy(res))
```

register_engine *Register a custom fitting engine*

Description

Registers a custom fitting engine for use with `loo_sensitivity`. The function receives the OmicsInput, the design matrix, and optional contrast/block arguments, and must return coefficient, standard-error, and p-value matrices with features as rows and model terms as columns.

Usage

```
register_engine(name, fit_fun)
```

Arguments

`name`	Engine id used in `loo_sensitivity(engine = ...)`.
`fit_fun`	function(input, design, contrast, block, ...) returning list(betas, se, p) matrices (features x terms).

Value

Invisibly returns the registered engine name.

Examples

```
example_fit <- function(input, design, contrast = NULL, block = NULL, ...) {
  features <- rownames(input@assay)
  terms <- colnames(design)
  estimates <- matrix(0, length(features), length(terms),
                      dimnames = list(features, terms))
  list(betas = estimates, se = estimates, p = estimates,
       colnames = terms)
}
register_engine("example_engine", example_fit)
rm(list = "example_engine",
   envir = getFromNamespace(".engine_registry", "looOmics"))
```

`run_engine` *Fit the full-sample model with a registered engine*

Description

Internal dispatch used by `loo_sensitivity`; also the extension point for custom engines together with `register_engine`. The method dispatches on the engine id stored on the input: built-in engines first, then registered custom engines.

Usage

```
run_engine(input, design, contrast = NULL, block = NULL, ...)
## S4 method for signature 'OmicsInput'
run_engine(input, design, contrast = NULL,
  block = NULL, ...)
```

Arguments

`input`	An `OmicsInput`.
`design`	Model matrix (rows are samples).
`contrast, block`	Optional contrast specification and blocking factor, passed through to the engine.
`...`	Engine-specific arguments.

Value

A list with elements betas, se, p (features x terms matrices) and colnames (model terms).

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(8))
x <- matrix(rnorm(10 * 8), nrow = 10,
            dimnames = list(paste0("gene", 1:10), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 4)),
                 row.names = samples)
input <- as_omics_input(x, ~ condition, col_data = cd)
design <- stats::model.matrix(~ condition, data = cd)
fit <- run_engine(input, design)
dim(fit$betas)
```

`tidy.LOOResult` *Tidy a LOOResult into a tibble*

Description

Extracts the shift table, influence scores, or flags as a tibble.

Usage

```
## S3 method for class 'LOOResult'
tidy(x, what = c("shifts", "influence", "flags"), ...)
```

Arguments

`x`	A `LOOResult`.
`what`	One of "shifts" (default), "influence", or "flags".
`...`	Unused.

Value

A tibble.

Examples

```
set.seed(1)
samples <- paste0("S", seq_len(12))
x <- matrix(rnorm(20 * 12), nrow = 20,
            dimnames = list(paste0("gene", 1:20), samples))
cd <- data.frame(condition = factor(rep(c("A", "B"), each = 6)),
                 row.names = samples)
res <- loo_sensitivity(x, ~ condition, col_data = cd, parallel = FALSE)
head(tidy(res, what = "influence"))
```

Index

∗ **package**
    looOmics-package, 2

as_omics_input, 3, *3*, *11*
as_omics_input,data.frame-method
        *(*as_omics_input*)*, 3
as_omics_input,ExpressionSet-method
        *(*as_omics_input*)*, 3
as_omics_input,matrix-method
        *(*as_omics_input*)*, 3
as_omics_input,phyloseq-method
        *(*as_omics_input*)*, 3
as_omics_input,SummarizedExperiment-method
        *(*as_omics_input*)*, 3
assay,OmicsInput,missing-method, 4
autoplot, *15*
autoplot *(*reexports*)*, 15
autoplot.LOOResult, 5, *15*

coef,LOOResult-method, 5
coef,LOOResult-method
        *(*LOOResult-methods*)*, 10

flag_influential, 6, *9*

influence_by_term, *3*, 7

loo_sensitivity, *3*, 8, *9*, *15, 16*
looOmics *(*looOmics-package*)*, 2
looOmics-package, 2
LOOResult, *3*, *5–8*, *10*, *13–15*, *17*
LOOResult *(*LOOResult-class*)*, 9
LOOResult-class, 9
LOOResult-methods, 10

OmicsInput, *3, 4*, *16*
OmicsInput *(*OmicsInput-class*)*, 11
OmicsInput-class, 11

plot,LOOResult,missing-method
        *(*autoplot.LOOResult*)*, 5
print.FlagProfile, 12
profile_flags, *9*, 12, *14*
profiles, *13*, 14

reexports, 15
register_engine, *8*, 15, *16*
run_engine, 16
run_engine,OmicsInput-method
        *(*run_engine*)*, 16

show,LOOResult-method
        *(*LOOResult-methods*)*, 10
summary,LOOResult-method
        *(*LOOResult-methods*)*, 10

tidy, *15*
tidy *(*reexports*)*, 15
tidy.LOOResult, *15*, 17