Package: MultiOmicsBridge (via r-universe)

November 25, 2026

Type Package

Title Integrative Multi-Omics Analysis of Host Transcriptomics and Gut Microbiome Data

Version 1.0.0

Description MultiOmicsBridge provides an end-to-end, reproducible computational framework for integrative analysis of paired host transcriptomics (bulk RNA-seq) and gut microbiome (16S rRNA or shotgun metagenomics) data. The package addresses the lack of a unified Bioconductor workflow for this pairing by implementing five modules: (1) data harmonization and normalization with CLR transformation for microbiome compositional data and TMM/voom for RNA-seq; (2) joint dimensionality reduction via sparse multi-block PLS-DA (DIABLO); (3) multi-omics biomarker discovery through cross-omics correlation networks and sparse feature loadings; (4) integrated diagnostic classification comparing host-only, microbiome-only, and joint Random Forest models with stratified cross-validation; and (5) publication-quality visualization of integration results, biomarker networks, classifier comparisons, and feature flow diagrams. All functions operate natively on SummarizedExperiment and MultiAssayExperiment objects and return a structured MOBResult S4 object. The package is validated on inflammatory bowel disease multi-omics data and designed with complex disease contexts (tuberculosis, HIV, EED) in mind.

License MIT + file LICENSE

URL https://github.com/SubhadipJana1409/MultiOmicsBridge

BugReports https://github.com/SubhadipJana1409/MultiOmicsBridge/issues

biocViews GeneExpression, Metagenomics, Classification, DimensionReduction, Normalization, StatisticalMethod, Sequencing, Microbiome, Transcriptomics, WorkflowStep, MultipleComparison, FeatureExtraction, Network, Visualization, QualityControl

Encoding UTF-8

Depends R (>= 4.5.0)

Imports SummarizedExperiment, MultiAssayExperiment, S4Vectors, BiocParallel, limma, edgeR, mixOmics, grid, methods, stats, utils, ggplot2, rlang, ranger, pROC, ggrepel

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), withr, curatedMetagenomicData, GEOquery

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libfreetype6-dev libglpk-dev libglu1-mesa-dev make texlive libicu-dev libpng-dev libuv1-dev libxml2-dev libgl1-mesa-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:25 UTC

RemoteUrl https://github.com/bioc/MultiOmicsBridge

RemoteRef RELEASE_3_24

RemoteSha b519a8d7985b8c1f7bcdd9a83d903286f9bd2d5b

Contents

MultiOmicsBridge-package . . . 3
biomarkerDiscovery . . . 4
biomarkers . . . 6
diagnosticClassifier . . . 7
example_multiomics . . . 9
featureLoadings . . . 10
generateReport . . . 11
integrationScores . . . 12
jointDimReduction . . . 13
loadHostData . . . 15
loadMicrobiomeData . . . 16
matchSamples . . . 18
MOBResult . . . 19
MOBResult-class . . . 21
MultiOmicsBridgeAnalysis . . . 21
performance . . . 23
plotBiomarkerNetwork . . . 24
plotClassifierComparison . . . 26
plotIntegration . . . 27
plotSankey . . . 28
show,MOBResult-method . . . 30

Index **31**

MultiOmicsBridge-package

MultiOmicsBridge: Integrative Multi-Omics Analysis of Host Transcriptomics and Gut Microbiome Data

Description

MultiOmicsBridge provides an end-to-end, reproducible computational framework for integrative analysis of paired host transcriptomics (bulk RNA-seq) and gut microbiome (16S rRNA or shotgun metagenomics) data. The package bridges two complementary biological data layers that, when analyzed together, reveal insights neither can provide alone.

The problem this package solves

No single Bioconductor package provides all four critical capabilities for host-microbiome integration in a unified, microbiome-aware workflow:

1. Proper normalization for **compositional microbiome data** alongside RNA-seq count data.
2. **Joint dimensionality reduction** identifying shared variation patterns between the two data layers.
3. **Multi-omics biomarker selection** identifying which host genes and which microbial taxa jointly predict disease status.
4. An **integrated diagnostic classifier** demonstrating the added value of combining both data types over either alone.

MultiOmicsBridge provides all four in a single, opinionated pipeline.

Five analysis modules

Module 1 — Data Harmonization `loadHostData`, `loadMicrobiomeData`, `matchSamples`: Import, normalize, and match paired omics data into a `MultiAssayExperiment`.

Module 2 — Joint Dimensionality Reduction `jointDimReduction`: Sparse multi-block PLS-DA (DIABLO) identifying correlated features across data blocks.

Module 3 — Biomarker Discovery `biomarkerDiscovery`: Sparse feature loadings and cross-omics correlation networks for multi-omics biomarker ranking.

Module 4 — Diagnostic Classification `diagnosticClassifier`: Host-only, microbiome-only, and joint Random Forest classifiers with stratified k-fold cross-validation.

Module 5 — Visualization and Reporting `plotIntegration`, `plotBiomarkerNetwork`, `plotClassifierComparison`, `plotSankey`, `generateReport`.

Main functions

`loadHostData` Import and voom-normalize bulk RNA-seq count data.

`loadMicrobiomeData` Import and CLR-transform microbiome taxa table data.

`matchSamples` Match paired samples across omics layers into a `MultiAssayExperiment`.

`jointDimReduction` Run DIABLO joint dimensionality reduction.

`biomarkerDiscovery` Identify ranked multi-omics biomarkers.

`diagnosticClassifier` Train and compare single-omics and joint diagnostic classifiers.

`MultiOmicsBridgeAnalysis` One-call wrapper for the complete analysis pipeline.

Complex disease contexts

The package is designed as a generalized framework and validated on tuberculosis, HIV antiretroviral therapy, and inflammatory bowel disease datasets. By providing a standardized, accessible workflow, MultiOmicsBridge lowers the barrier to multi-omics integration for researchers working across various disease contexts.

Author(s)

Maintainer: Subhadip Jana <subhadipjana1409@gmail.com> (ORCID) [funder]

Authors:

- Subhadip Jana <subhadipjana1409@gmail.com> (ORCID) [funder]

References

Rohart F et al. (2017). mixOmics: An R package for 'omics feature selection and multiple data integration. *PLoS Comput Biol*, 13(11), e1005752.

Reel PS et al. (2021). Using machine learning approaches for multi-omics data analysis: A review. *Biotechnol Adv*, 49, 107739.

Franzosa EA et al. (2019). Gut microbiome structure and metabolic activity in inflammatory bowel disease. *Nature Microbiology*, 4, 293-305.

See Also

Useful links:

- https://github.com/SubhadipJana1409/MultiOmicsBridge
- Report bugs at https://github.com/SubhadipJana1409/MultiOmicsBridge/issues

---

`biomarkerDiscovery` *Multi-Omics Biomarker Discovery*

Description

Identifies and ranks host genes and microbial taxa that jointly predict the outcome of interest using two complementary evidence streams: (1) sparse feature loadings from the DIABLO joint dimensionality reduction model, and (2) a cross-omics Spearman correlation network linking host genes to microbial taxa. Features are ranked by their combined loading score and annotated with their maximum cross-omics correlation.

Usage

```
biomarkerDiscovery(
  mae,
  dr_result,
  n_biomarkers = 50L,
  host_assay = "voom",
  mb_assay = "CLR"
)
```

Arguments

mae	A MultiAssayExperiment from matchSamples.
dr_result	A named list from jointDimReduction.
n_biomarkers	An integer(1) number of top features to select per omics layer. Default: 50.
host_assay	A character(1) assay in host SE for computing cross-omics correlations. Default: "voom".
mb_assay	A character(1) assay in microbiome SE for computing cross-omics correlations. Default: "CLR".

Details

The biomarker ranking combines:

Sparse loading score The L2 norm of a feature's loadings across all DIABLO components. Genes/taxa with higher loading scores contribute more strongly to the latent integration axes.

Cross-omics correlation For each selected host gene, the maximum absolute Spearman correlation with any selected microbial taxon (and vice versa). High cross-omics correlation indicates biologically relevant host-microbe co-variation.

Hub features — those with both high loading scores and high cross-omics correlations — represent the most credible multi-omics biomarker candidates.

Value

A DataFrame with one row per biomarker and columns:

feature Feature name (gene or taxon ID).

omics_layer Either "host" or "microbiome".

loading_score L2 norm of DIABLO loadings across components.

rank Within-layer ranking by loading score.

component DIABLO component with highest absolute loading.

max_cross_cor Maximum absolute Spearman correlation with a feature from the other omics layer.

top_partner Name of the cross-omics feature with the highest absolute correlation.

See Also

jointDimReduction, plotBiomarkerNetwork, MultiOmicsBridgeAnalysis

Examples

```
set.seed(42)
host_counts <- matrix(rpois(500 * 20, 100), nrow = 500, ncol = 20,
    dimnames = list(paste0("Gene", 1:500), paste0("S", 1:20)))
host_counts[1:20, 11:20] <- host_counts[1:20, 11:20] * 5L
mb_counts <- matrix(rpois(60 * 20, 40), nrow = 60, ncol = 20,
    dimnames = list(paste0("Taxon", 1:60), paste0("S", 1:20)))

host_se <- loadHostData(host_counts)
mb_se   <- loadMicrobiomeData(mb_counts)
mae     <- matchSamples(host_se, mb_se)
outcome <- rep(c("ctrl", "treat"), each = 10)
dr_res  <- jointDimReduction(mae, outcome, n_components = 2,
                             n_features_host = 30, n_features_mb = 15)
bm <- biomarkerDiscovery(mae, dr_res, n_biomarkers = 20)
head(as.data.frame(bm))
```

---

biomarkers *Accessor for biomarker table in a MOBResult*

Description

Returns the ranked multi-omics biomarker DataFrame from a MOBResult object.

Usage

```
biomarkers(x, ...)

## S4 method for signature 'MOBResult'
biomarkers(x, ...)
```

Arguments

x	A MOBResult object.
...	Additional arguments (not used).

Value

A DataFrame with columns feature, omics_layer, loading_score, rank, and component.

Examples

```
library(S4Vectors)
bm <- DataFrame(feature = c("G1","T1"), omics_layer = c("host","microbiome"),
                loading_score = c(0.8,0.6), rank = c(1L,2L), component = c(1L,1L))
obj <- MOBResult(matrix(rnorm(20), 10, 2), list(), bm, list())
biomarkers(obj)
```

---

diagnosticClassifier *Integrated Diagnostic Classifier with Multi-Omics Comparison*

Description

Trains and evaluates three Random Forest diagnostic classifiers using cross-validation: a host-only model, a microbiome-only model, and a joint multi-omics model. By comparing AUC-ROC across all three configurations, `diagnosticClassifier` quantifies the added diagnostic value of combining both data types.

Usage

```
diagnosticClassifier(
  mae,
  outcome,
  biomarker_table = NULL,
  cv_folds = 5L,
  n_trees = 500L,
  seed = 42L,
  host_assay = "voom",
  mb_assay = "CLR"
)
```

Arguments

`mae`	A `MultiAssayExperiment` from `matchSamples`.
`outcome`	A `character` or `factor` vector of outcome labels, one per sample. Must have exactly 2 levels.
`biomarker_table`	An optional `DataFrame` from `biomarkerDiscovery`. If provided, only the listed features are used as classifier inputs. If `NULL`, all features in the matched MAE are used. Default: `NULL`.
`cv_folds`	An `integer(1)` number of cross-validation folds. Default: `5`.
`n_trees`	An `integer(1)` number of trees in each Random Forest. Default: `500`.
`seed`	An `integer(1)` random seed for reproducibility. Default: `42`.
`host_assay`	A `character(1)` assay in the host SE. Default: `"voom"`.
`mb_assay`	A `character(1)` assay in the microbiome SE. Default: `"CLR"`.

Details

Each classifier is trained using `ranger::ranger` (fast C++ Random Forest) with 500 trees and stratified k-fold cross-validation. Features are drawn from the top biomarkers identified by `biomarkerDiscovery` (or all available features if `biomarker_table` is `NULL`).

The cross-validation procedure:

1. Split samples into `cv_folds` stratified folds (each fold preserves the outcome class ratio).
2. For each fold, train on the remaining folds, predict on the held-out fold.
3. Compute AUC-ROC on held-out predictions.
4. Report mean +/- SD AUC across folds.

Value

A named `list` with elements:

`host_only` List with `mean_auc`, `sd_auc`, `fold_auc`, `roc_data` for the host-only classifier.

`microbiome_only` Same structure for the microbiome-only classifier.

`joint` Same structure for the joint classifier.

`n_features` Named integer: `host`, `microbiome`, `joint` feature counts.

`cv_folds` Number of CV folds used.

`outcome_levels` Outcome levels.

See Also

`biomarkerDiscovery`, `plotClassifierComparison`, `MultiOmicsBridgeAnalysis`

Examples

```
set.seed(42)
host_counts <- matrix(rpois(500 * 20, 100), nrow = 500, ncol = 20,
    dimnames = list(paste0("Gene", 1:500), paste0("S", 1:20)))
host_counts[1:20, 11:20] <- host_counts[1:20, 11:20] * 5L
mb_counts <- matrix(rpois(60 * 20, 40), nrow = 60, ncol = 20,
    dimnames = list(paste0("Taxon", 1:60), paste0("S", 1:20)))

host_se <- loadHostData(host_counts)
mb_se   <- loadMicrobiomeData(mb_counts)
mae     <- matchSamples(host_se, mb_se)
outcome <- rep(c("ctrl", "treat"), each = 10)
dr_res  <- jointDimReduction(mae, outcome, n_components = 2,
                             n_features_host = 30, n_features_mb = 15)
bm      <- biomarkerDiscovery(mae, dr_res, n_biomarkers = 20)

clf_res <- diagnosticClassifier(mae, outcome,
                                biomarker_table = bm, cv_folds = 3)
clf_res$host_only$mean_auc
clf_res$joint$mean_auc
```

`example_multiomics` *Example Multi-Omics Dataset for MultiOmicsBridge*

Description

A small, self-contained simulated paired multi-omics dataset inspired by inflammatory bowel disease (IBD) biology. This dataset is used throughout package examples, tests, and the vignette to demonstrate the complete MultiOmicsBridge analysis pipeline.

Format

A named list with 6 elements. See Details.

Details

The dataset is a named list stored as an RDS file in `inst/extdata/example_multiomics.rds`. It contains 30 samples (15 ulcerative colitis [UC], 15 healthy controls) with paired host transcriptomics (800 genes) and gut microbiome (60 taxa) measurements.

Data provenance: All data are computationally simulated (not derived from real patient data). Host RNA-seq counts are drawn from a negative binomial distribution (`rnbinom(mu=180, size=3)`) with IBD-associated fold changes injected for 25 signal genes. Microbiome counts are drawn from `rnbinom(mu=600, size=2)` with dysbiosis fold changes for 15 signal taxa. A deliberate cross-omics correlation between CXCL8 (host) and Ruminococcus_gnavus (microbiome) is injected to test biomarker network recovery. The random seed is `set.seed(2026L)`.

List elements:

`host_se` A `SummarizedExperiment` with 800 genes (rows) x 30 samples (columns). Contains raw counts and voom-normalized assays. Includes 15 IBD-upregulated genes (CXCL8, S100A8, IL6, TNF, etc.), 10 IBD-downregulated genes (MUC2, TFF3, etc.), and 775 neutral background genes.

`mb_se` A `SummarizedExperiment` with 60 taxa (rows) x 30 samples (columns). Contains raw counts and CLR-normalized assays. Includes 8 UC-enriched taxa (Ruminococcus_gnavus, Escherichia_coli, etc.), 7 UC-depleted taxa (Faecalibacterium_prausnitzii, Roseburia_intestinalis, etc.), and 45 neutral background taxa.

`mae` A `MultiAssayExperiment` combining both omics layers with matched sample mappings.

`outcome` A character vector of length 30 with values `"UC"` (first 15) and `"Control"` (last 15).

`metadata` A `data.frame` with 30 rows and columns: `condition`, `age`, `sex`, `bmi`, `disease_duration_yrs` (NA for controls).

`ground_truth` A list containing known signal features for validation: `ibd_up_genes` (15 gene names), `ibd_down_genes` (10 gene names), `uc_up_taxa` (8 taxon names), `uc_down_taxa` (7 taxon names), and `correlated_pair` (CXCL8 and Ruminococcus_gnavus).

Value

A named list with 6 elements: `host_se`, `mb_se`, `mae`, `outcome`, `metadata`, and `ground_truth`. See Details for full descriptions of each element.

Source

Generated by `inst/scripts/make_example_data.R`. All data are simulated; no real patient data is included.

See Also

`MultiOmicsBridgeAnalysis`, `loadHostData`, `loadMicrobiomeData`, `matchSamples`

Examples

```
ex <- readRDS(system.file("extdata", "example_multiomics.rds",
                          package = "MultiOmicsBridge"))
names(ex)
ex$mae
table(ex$outcome)
head(ex$metadata)
```

---

featureLoadings *Accessor for feature loadings in a MOBResult*

Description

Returns the named list of per-layer feature loading matrices from a `MOBResult` object.

Usage

```
featureLoadings(x, ...)

## S4 method for signature 'MOBResult'
featureLoadings(x, ...)
```

Arguments

`x`	A `MOBResult` object.
`...`	Additional arguments (not used).

Value

A named `list` with elements `host` and `microbiome`, each a matrix of genes/taxa x components.

Examples

```
library(S4Vectors)
fl <- list(host = matrix(rnorm(10), 5, 2), microbiome = matrix(rnorm(6), 3, 2))
obj <- MOBResult(matrix(rnorm(20), 10, 2), fl, DataFrame(), list())
featureLoadings(obj)
```

generateReport *Generate a Structured Summary Report of MultiOmicsBridge Results*

Description

Prints a formatted text summary of a MOBResult object to the console and optionally saves it to a plain-text file. The report covers all five analysis modules: data dimensions, integration method, top biomarkers, cross-omics correlations, and classifier performance comparison.

For a full interactive HTML report, users can render the package vignette template (`system.file("vignettes", package = "MultiOmicsBridge")`) with `rmarkdown::render()` using their own `MOBResult` object.

Usage

```
generateReport(result, file = NULL, n_top = 10L)
```

Arguments

result	A MOBResult object from MultiOmicsBridgeAnalysis.
file	An optional `character(1)` file path where the report should be saved as a plain text file. If `NULL` (default), the report is only printed to the console.
n_top	An `integer(1)` number of top biomarkers to list. Default: `10`.

Value

Invisibly returns a named `list` of character vectors, one per report section.

See Also

MultiOmicsBridgeAnalysis, MOBResult

Examples

```
library(S4Vectors)
scores <- matrix(rnorm(20), nrow = 10, ncol = 2,
    dimnames = list(paste0("S", 1:10), c("Comp1", "Comp2")))
bm <- DataFrame(
    feature = c("Gene1", "Taxon1", "Gene2"),
    omics_layer = c("host", "microbiome", "host"),
    loading_score = c(0.9, 0.8, 0.7),
    rank = c(1L, 2L, 3L), component = c(1L, 1L, 1L)
)
cr <- list(
    host_only       = list(mean_auc = 0.82, sd_auc = 0.05,
                           fold_auc = c(0.78, 0.84, 0.83)),
    microbiome_only = list(mean_auc = 0.75, sd_auc = 0.06,
                           fold_auc = c(0.70, 0.79, 0.76)),
    joint           = list(mean_auc = 0.94, sd_auc = 0.03,
```

```
                          fold_auc = c(0.92, 0.95, 0.94)),
    cv_folds        = 3L,
    outcome_levels  = c("ctrl", "treat")
)
obj <- MOBResult(scores, list(), bm, cr,
    params = list(integration_method = "DIABLO",
                  n_components = 2L,
                  outcome_levels = c("ctrl", "treat"),
                  cv_folds = 3L))
generateReport(obj, n_top = 3)
```

`integrationScores` *Accessor for integrated sample scores in a MOBResult*

Description

Returns the matrix of integrated sample scores (samples x latent components) from a `MOBResult` object.

Usage

```
integrationScores(x, ...)

## S4 method for signature 'MOBResult'
integrationScores(x, ...)
```

Arguments

`x`	A `MOBResult` object.
`...`	Additional arguments (not used).

Value

A `matrix` with rows = samples and columns = latent components.

Examples

```
library(S4Vectors)
scores <- matrix(rnorm(20), nrow = 10, ncol = 2,
                 dimnames = list(paste0("S",1:10), c("Comp1","Comp2")))
obj <- MOBResult(scores, list(), DataFrame(), list())
integrationScores(obj)
```

`jointDimReduction` *Joint Dimensionality Reduction via Sparse Multi-Block PLS-DA*

Description

Performs joint dimensionality reduction across host transcriptomics and gut microbiome data using DIABLO (Data Integration Analysis for Biomarker discovery using Latent cOmponents), a sparse multi-block PLS-DA implemented in `mixOmics`. DIABLO simultaneously identifies correlated features across data blocks while discriminating between outcome groups, enforcing sparsity so only the most informative features contribute to each latent component.

Usage

```
jointDimReduction(
  mae,
  outcome,
  n_components = 2L,
  n_features_host = 50L,
  n_features_mb = 20L,
  design_off_diag = 0.1,
  host_assay = "voom",
  mb_assay = "CLR",
  min_variance = 0
)
```

Arguments

`mae`	A `MultiAssayExperiment` from `matchSamples` containing `"host"` and `"microbiome"` experiments.
`outcome`	A `character` or `factor` vector of outcome labels, one per sample (in the same order as `colnames(mae)`). Must have exactly 2 or more levels.
`n_components`	An `integer(1)` number of latent components to extract. Default: `2`.
`n_features_host`	An `integer(1)` number of host genes to retain per component (sparse selection). Default: `50`.
`n_features_mb`	An `integer(1)` number of microbial taxa to retain per component. Default: `20`.
`design_off_diag`	A `numeric(1)` in `[0, 1]` controlling the off-diagonal entries of the DIABLO design matrix. Higher values emphasize cross-block correlation; lower values prioritize outcome discrimination. Default: `0.1`.
`host_assay`	A `character(1)` name of the assay in the host `SummarizedExperiment` to use as input. Default: `"voom"`.
`mb_assay`	A `character(1)` name of the assay in the microbiome `SummarizedExperiment` to use as input. Default: `"CLR"`.
`min_variance`	A `numeric(1)` minimum variance (as a fraction of total row variance) for genes/taxa to be retained before DIABLO. Default: `0` (no filtering).

Details

The DIABLO model is fitted as:

$$\max \text{Cov}(t_{\text{host}}, t_{\text{mb}})$$

subject to $||w_{\text{host}}||_1 \leq \lambda_{\text{host}}$ and $||w_{\text{mb}}||_1 \leq \lambda_{\text{mb}}$, where t are the sample scores (variates) and w are the sparse feature weights (loadings).

The number of features retained per component is controlled by `n_features_host` and `n_features_mb`. By default, a design matrix connecting all blocks with moderate correlation (`design_off_diag = 0.1`) is used, which prioritizes outcome discrimination over cross-block correlation.

Value

A named `list` with elements:

`scores` A `matrix` (samples x components) of integrated sample scores (from the host block variate).

`host_loadings` A `matrix` (genes x components) of sparse host feature loadings.

`mb_loadings` A `matrix` (taxa x components) of sparse microbiome feature loadings.

`explained_variance` A named `numeric` vector of explained variance per component.

`diablo_object` The full DIABLO result object from `mixOmics::block.splsda`.

`outcome` The outcome factor used.

See Also

`biomarkerDiscovery`, `plotIntegration`, `MultiOmicsBridgeAnalysis`

Examples

```
set.seed(42)
host_counts <- matrix(rpois(500 * 20, 100), nrow = 500, ncol = 20,
    dimnames = list(paste0("Gene", 1:500), paste0("S", 1:20)))
host_counts[1:20, 11:20] <- host_counts[1:20, 11:20] * 5L
mb_counts <- matrix(rpois(60 * 20, 40), nrow = 60, ncol = 20,
    dimnames = list(paste0("Taxon", 1:60), paste0("S", 1:20)))

host_se <- loadHostData(host_counts)
mb_se   <- loadMicrobiomeData(mb_counts)
mae     <- matchSamples(host_se, mb_se)

outcome <- rep(c("ctrl", "treat"), each = 10)
dr_res  <- jointDimReduction(mae, outcome = outcome,
                             n_components = 2,
                             n_features_host = 30,
                             n_features_mb   = 15)
dim(dr_res$scores)
head(dr_res$explained_variance)
```

loadHostData *Load and Normalize Host Bulk RNA-seq Data*

Description

Imports a bulk RNA-seq count matrix and applies TMM normalization followed by limma-voom precision weighting. The result is a SummarizedExperiment with both the raw counts and voom-transformed log2-CPM values stored as named assays.

Usage

```
loadHostData(counts, col_data = NULL, min_count = 1, assay_name = "voom")
```

Arguments

counts	A matrix of non-negative integer counts with genes in rows and samples in columns. Alternatively, a SummarizedExperiment whose "counts" assay is used.
col_data	An optional data.frame or DataFrame of sample-level metadata. Row names must match the column names of counts. If NULL, an empty DataFrame is used. Default: NULL.
min_count	A numeric(1) minimum total count per gene. Genes with row sums below this threshold are removed before normalization. Default: 1 (removes all-zero genes only).
assay_name	A character(1) name for the voom-normalized assay in the output SummarizedExperiment. Default: "voom".

Details

The normalization pipeline:

1. Construct a DGEList from raw counts.
2. Apply TMM (trimmed mean of M-values) normalization via edgeR::normLibSizes to remove compositional bias between libraries.
3. Apply limma::voom to compute log2-CPM values with precision weights that model the mean-variance trend. These weights are used downstream by diagnosticClassifier and jointDimReduction.

Genes with very low expression are not filtered here; gene filtering is performed within jointDimReduction and diagnosticClassifier based on expression in the matched multi-omics dataset. Users may pre-filter genes if desired.

Value

A `SummarizedExperiment` with two assays:

`"counts"` Raw integer count matrix.

`"voom"` Voom-transformed log2-CPM matrix with TMM library size normalization applied.

Sample metadata (if provided) is stored in `colData`.

See Also

`loadMicrobiomeData`, `matchSamples`, `MultiOmicsBridgeAnalysis`

Examples

```
set.seed(42)
n_genes   <- 200
n_samples <- 20
counts <- matrix(rpois(n_genes * n_samples, lambda = 150),
                 nrow = n_genes, ncol = n_samples)
rownames(counts) <- paste0("Gene", seq_len(n_genes))
colnames(counts) <- paste0("Sample", seq_len(n_samples))

col_data <- data.frame(
    condition = rep(c("ctrl", "treat"), each = 10),
    row.names = colnames(counts)
)

host_se <- loadHostData(counts, col_data = col_data)
host_se
SummarizedExperiment::assayNames(host_se)
```

`loadMicrobiomeData` *Load and Normalize Microbiome Taxa Table Data*

Description

Imports a microbiome taxa count table and applies either centered log-ratio (CLR) or total sum scaling (TSS) normalization. The result is a `SummarizedExperiment` with both raw counts and normalized values stored as named assays.

Usage

```
loadMicrobiomeData(
  taxa_table,
  col_data = NULL,
  normalization = c("CLR", "TSS"),
  pseudocount = 0.5,
  min_prevalence = 0.1
)
```

Arguments

`taxa_table`	A `matrix` of non-negative integer counts. Expected orientation: taxa in **rows**, samples in **columns** (Bioconductor convention). If samples appear to be in rows (more rows than columns and row names suggest samples), the matrix is transposed with a warning.
`col_data`	An optional `data.frame` or `DataFrame` of sample-level metadata. Row names must match the column names of `taxa_table`. Default: `NULL`.
`normalization`	A `character(1)`, either `"CLR"` (centered log-ratio, default) or `"TSS"` (total sum scaling).
`pseudocount`	A positive `numeric(1)` pseudocount added to all counts before log-transformation (CLR only). Default: `0.5`.
`min_prevalence`	A `numeric(1)` in `[0, 1]`. Taxa present in fewer than this fraction of samples are removed. Default: `0.1` (remove taxa in fewer than 10% of samples).

Details

Microbiome data is **compositional**: only relative abundances are observed, not absolute counts. Treating compositional data with standard correlation or distance measures leads to spurious results (the Aitchison problem). MultiOmicsBridge applies one of two microbiome-appropriate normalizations:

CLR (default) The centered log-ratio transformation:

$$clr(x_j) = \log(x_j + \delta) - \frac{1}{D}\sum_{k=1}^{D}\log(x_k + \delta)$$

where δ is the pseudocount and the sum is over all D taxa. CLR maps compositional data to real space and removes the unit-sum constraint, enabling Euclidean geometry.

TSS Total sum scaling divides each sample by its library size, producing relative abundances (proportions). Simpler but retains the compositional constraint.

Zero counts are handled by adding a small pseudocount before log-transformation; the default pseudocount of 0.5 is a conservative choice appropriate for sparse 16S data.

Value

A `SummarizedExperiment` with two assays:

`"counts"` Raw integer count matrix (taxa x samples).

`"CLR"` **or** `"TSS"` Normalized values.

See Also

`loadHostData`, `matchSamples`, `MultiOmicsBridgeAnalysis`

Examples

```
set.seed(42)
n_taxa    <- 80
n_samples <- 20
taxa_table <- matrix(rpois(n_taxa * n_samples, lambda = 30),
                     nrow = n_taxa, ncol = n_samples)
rownames(taxa_table) <- paste0("Taxon", seq_len(n_taxa))
colnames(taxa_table) <- paste0("Sample", seq_len(n_samples))

mb_se <- loadMicrobiomeData(taxa_table, normalization = "CLR")
mb_se
SummarizedExperiment::assayNames(mb_se)
```

`matchSamples` *Match Paired Samples Across Host and Microbiome Datasets*

Description

Identifies samples present in both host and microbiome `SummarizedExperiment` objects, subsets both to the common samples, and assembles a `MultiAssayExperiment` (MAE) that serves as the primary input for downstream analysis functions.

Usage

```
matchSamples(
  host_se,
  mb_se,
  sample_col_host = NULL,
  sample_col_mb = NULL,
  min_paired = 5L
)
```

Arguments

`host_se`	A `SummarizedExperiment` from `loadHostData`, with samples as columns.
`mb_se`	A `SummarizedExperiment` from `loadMicrobiomeData`, with samples as columns.
`sample_col_host`	A `character(1)` naming the column in `colData(host_se)` that contains the sample identifier. If `NULL` (default), uses `colnames(host_se)`.
`sample_col_mb`	A `character(1)` naming the column in `colData(mb_se)` that contains the sample identifier. If `NULL` (default), uses `colnames(mb_se)`.
`min_paired`	An `integer(1)` minimum number of paired samples required to proceed. Default: 5.

Details

In paired multi-omics studies, not all samples necessarily have both data types due to sequencing failures, QC exclusions, or study design. `matchSamples` transparently reports how many samples are retained and warns if fewer than `min_paired` paired samples are found.

The output `MultiAssayExperiment` stores the host and microbiome `SummarizedExperiment` objects under the names `"host"` and `"microbiome"` respectively, with a unified `colData` drawn from the host sample metadata.

Value

A `MultiAssayExperiment` with two experiments:

`"host"` Subset of `host_se` for paired samples.

`"microbiome"` Subset of `mb_se` for paired samples.

The `colData` of the MAE is taken from `host_se` for the paired samples.

See Also

`loadHostData`, `loadMicrobiomeData`, `MultiOmicsBridgeAnalysis`

Examples

```
set.seed(42)
# Host data: 200 genes, 20 samples
host_counts <- matrix(rpois(200 * 20, 150),
                      nrow = 200, ncol = 20,
                      dimnames = list(paste0("Gene", 1:200),
                                      paste0("Sample", 1:20)))
host_se <- loadHostData(host_counts)

# Microbiome data: 50 taxa, 18 samples (2 missing)
mb_counts <- matrix(rpois(50 * 18, 30),
                    nrow = 50, ncol = 18,
                    dimnames = list(paste0("Taxon", 1:50),
                                    paste0("Sample", 1:18)))
mb_se <- loadMicrobiomeData(mb_counts)

mae <- matchSamples(host_se, mb_se, min_paired = 5)
mae
```

`MOBResult` *Constructor for MOBResult*

Description

Create a new `MOBResult` object.

Usage

```
MOBResult(
  integratedScores,
  featureLoadings,
  biomarkerTable,
  classifierResults,
  params = list()
)
```

Arguments

integratedScores	A matrix of integrated sample scores (samples x components).
featureLoadings	A named list with host and microbiome loading matrices.
biomarkerTable	A DataFrame of ranked multi-omics biomarkers.
classifierResults	A named list of classifier performance metrics.
params	A list of analysis parameters.

Value

A MOBResult object.

Examples

```
library(S4Vectors)
scores <- matrix(rnorm(20), nrow = 10, ncol = 2,
                 dimnames = list(paste0("S", 1:10), c("Comp1","Comp2")))
loadings <- list(
    host       = matrix(rnorm(10), nrow = 5, ncol = 2,
                        dimnames = list(paste0("G", 1:5), c("Comp1","Comp2"))),
    microbiome = matrix(rnorm(6), nrow = 3, ncol = 2,
                        dimnames = list(paste0("T", 1:3), c("Comp1","Comp2")))
)
bm <- DataFrame(
    feature      = c("G1", "T1"),
    omics_layer  = c("host", "microbiome"),
    loading_score = c(0.8, 0.6),
    rank         = c(1L, 2L),
    component    = c(1L, 1L)
)
obj <- MOBResult(
    integratedScores  = scores,
    featureLoadings   = loadings,
    biomarkerTable    = bm,
    classifierResults = list(),
    params = list(integration_method = "DIABLO")
)
obj
```

MOBResult-class *MOBResult: Multi-Omics Bridge Result Container*

Description

An S4 class storing the output of `MultiOmicsBridgeAnalysis`. Slots hold integrated sample scores from joint dimensionality reduction, per-layer feature loadings, a ranked multi-omics biomarker table, diagnostic classifier performance metrics, and analysis parameters.

Slots

`integratedScores` A `matrix` of integrated sample scores with rows = samples and columns = latent components. Produced by DIABLO.

`featureLoadings` A named `list` with two elements: `host` (genes x components) and `microbiome` (taxa x components) containing the sparse feature loading matrices.

`biomarkerTable` A `DataFrame` with one row per selected biomarker containing columns `feature`, `omics_layer`, `loading_score`, `rank`, and `component`.

`classifierResults` A named `list` with elements `host_only`, `microbiome_only`, and `joint`, each containing cross-validated AUC-ROC and fold-level performance metrics.

`params` A `list` of analysis parameters including `integration_method`, `n_components`, `n_biomarkers`, `cv_folds`, and `outcome_levels`.

MultiOmicsBridgeAnalysis *Full Multi-Omics Integration Analysis Pipeline*

Description

A one-call wrapper that executes the complete MultiOmicsBridge analysis pipeline: joint dimensionality reduction, multi-omics biomarker discovery, and integrated diagnostic classification. The function accepts a `MultiAssayExperiment` (from `matchSamples`) and returns a `MOBResult` S4 object containing all results.

Usage

```
MultiOmicsBridgeAnalysis(
  mae,
  outcome,
  n_components = 2L,
  n_features_host = 50L,
  n_features_mb = 20L,
  n_biomarkers = 50L,
```

```
  cv_folds = 5L,
  host_assay = "voom",
  mb_assay = "CLR",
  design_off_diag = 0.1,
  seed = 42L,
  BPPARAM = SerialParam()
)
```

Arguments

mae	A MultiAssayExperiment with "host" and "microbiome" experiments, produced by matchSamples.
outcome	A character or factor vector of outcome labels, one per sample (ordered as colnames(mae)). Must have exactly 2 levels for classification.
n_components	An integer(1) number of DIABLO latent components to extract. Default: 2.
n_features_host	An integer(1) number of host genes to select per DIABLO component. Default: 50.
n_features_mb	An integer(1) number of microbial taxa to select per DIABLO component. Default: 20.
n_biomarkers	An integer(1) total number of biomarkers to report per omics layer in the ranked table. Default: 50.
cv_folds	An integer(1) number of cross-validation folds for classifier evaluation. Default: 5.
host_assay	A character(1) name of the normalized host assay in the MAE. Default: "voom".
mb_assay	A character(1) name of the normalized microbiome assay in the MAE. Default: "CLR".
design_off_diag	A numeric(1) off-diagonal value for the DIABLO design matrix (see jointDimReduction). Default: 0.1.
seed	An integer(1) random seed. Default: 42.
BPPARAM	A BiocParallelParam object. Default: SerialParam().

Details

Internally, MultiOmicsBridgeAnalysis calls:

1. jointDimReduction — DIABLO sparse multi-block PLS-DA.
2. biomarkerDiscovery — loading-based ranking and cross-omics correlation annotation.
3. diagnosticClassifier — host-only, microbiome-only, and joint Random Forest classifiers with cross-validation.

Each step can also be called independently for fine-grained control.

Value

A MOBResult object with:

`integrationScores(result)` Matrix of DIABLO sample scores (samples x components).

`featureLoadings(result)` Named list of host and microbiome loading matrices.

`biomarkers(result)` Ranked multi-omics biomarker DataFrame.

`performance(result)` Classifier performance list with AUC-ROC for host-only, microbiome-only, and joint models.

See Also

matchSamples, jointDimReduction, biomarkerDiscovery, diagnosticClassifier, plotIntegration, plotClassifierComparison

Examples

```
set.seed(42)
host_counts <- matrix(rpois(500 * 20, 100), nrow = 500, ncol = 20,
    dimnames = list(paste0("Gene", 1:500), paste0("S", 1:20)))
host_counts[1:20, 11:20] <- host_counts[1:20, 11:20] * 5L
mb_counts <- matrix(rpois(60 * 20, 40), nrow = 60, ncol = 20,
    dimnames = list(paste0("Taxon", 1:60), paste0("S", 1:20)))

host_se <- loadHostData(host_counts)
mb_se   <- loadMicrobiomeData(mb_counts)
mae     <- matchSamples(host_se, mb_se)
outcome <- rep(c("ctrl", "treat"), each = 10)

result <- MultiOmicsBridgeAnalysis(mae, outcome,
                                   n_components    = 2,
                                   n_features_host = 20,
                                   n_features_mb   = 10,
                                   n_biomarkers    = 15,
                                   cv_folds        = 3)
result
head(as.data.frame(biomarkers(result)))
```

---

performance *Accessor for classifier performance in a MOBResult*

Description

Returns the cross-validated classifier performance metrics from a MOBResult object.

Usage

```
performance(x, ...)

## S4 method for signature 'MOBResult'
performance(x, ...)
```

Arguments

x	A MOBResult object.
...	Additional arguments (not used).

Value

A named list with elements host_only, microbiome_only, and joint, each containing mean_auc, sd_auc, and fold_auc.

Examples

```
library(S4Vectors)
cr <- list(host_only = list(mean_auc = 0.85),
           microbiome_only = list(mean_auc = 0.78),
           joint = list(mean_auc = 0.92))
obj <- MOBResult(matrix(rnorm(20), 10, 2), list(),
                 DataFrame(), cr)
performance(obj)
```

---

plotBiomarkerNetwork *Cross-Omics Biomarker Correlation Heatmap*

Description

Produces a clustered heatmap of Spearman correlations between the top selected host genes and microbial taxa. Rows represent host genes, columns represent microbial taxa, and cell colour encodes the Spearman correlation value. Strong positive or negative correlations between a host gene and a microbial taxon suggest a potential functional host-microbe interaction.

Usage

```
plotBiomarkerNetwork(
  result,
  mae,
  n_host = 20L,
  n_mb = 15L,
  host_assay = "voom",
  mb_assay = "CLR",
  cor_thresh = 0,
```

```
  low_colour = "#F44336",
  mid_colour = "white",
  high_colour = "#2196F3"
)
```

Arguments

result	A MOBResult object from MultiOmicsBridgeAnalysis.
mae	A MultiAssayExperiment from matchSamples, used to compute correlations from assay data.
n_host	An integer(1) number of top host genes to display. Default: 20.
n_mb	An integer(1) number of top microbial taxa to display. Default: 15.
host_assay	A character(1) assay name in host SE. Default: "voom".
mb_assay	A character(1) assay name in microbiome SE. Default: "CLR".
cor_thresh	A numeric(1) minimum absolute correlation to display (cells below this threshold are shown in grey). Default: 0.
low_colour	A character(1) colour for strong negative correlations. Default: "#F44336".
mid_colour	A character(1) colour for zero correlation. Default: "white".
high_colour	A character(1) colour for strong positive correlations. Default: "#2196F3".

Value

A ggplot2 object.

See Also

MultiOmicsBridgeAnalysis, plotIntegration

Examples

```
set.seed(42)
host_counts <- matrix(rpois(500 * 20, 100), nrow = 500, ncol = 20,
    dimnames = list(paste0("Gene", 1:500), paste0("S", 1:20)))
host_counts[1:20, 11:20] <- host_counts[1:20, 11:20] * 5L
mb_counts <- matrix(rpois(60 * 20, 40), nrow = 60, ncol = 20,
    dimnames = list(paste0("Taxon", 1:60), paste0("S", 1:20)))

host_se <- loadHostData(host_counts)
mb_se   <- loadMicrobiomeData(mb_counts)
mae     <- matchSamples(host_se, mb_se)
outcome <- rep(c("ctrl", "treat"), each = 10)
result  <- MultiOmicsBridgeAnalysis(mae, outcome,
                                    n_components = 2,
                                    n_features_host = 20,
                                    n_features_mb = 10,
                                    n_biomarkers = 15, cv_folds = 3)
plotBiomarkerNetwork(result, mae, n_host = 10, n_mb = 8)
```

plotClassifierComparison *ROC Curve Comparison of Host-Only, Microbiome-Only, and Joint Classifiers*

Description

Produces overlaid ROC (Receiver Operating Characteristic) curves comparing the three diagnostic classifier configurations — host transcriptomics only, gut microbiome only, and the joint multi-omics model — on the same axes. The AUC-ROC values are annotated on the plot, making the multi-omics advantage immediately visible. A bar chart of mean cross-validated AUC values is also available via `type = "bar"`.

Usage

```
plotClassifierComparison(
  result,
  type = c("roc", "bar"),
 colours = c(host_only = "#4CAF50", microbiome_only = "#FF9800", joint = "#2196F3"),
  show_diagonal = TRUE,
  linewidth = 0.9
)
```

Arguments

`result`	A `MOBResult` object from `MultiOmicsBridgeAnalysis`.
`type`	A `character(1)`, either `"roc"` (ROC curves from the last CV fold) or `"bar"` (bar chart of mean CV AUC values). Default: `"roc"`.
`colours`	A named `character` vector of colours for `"host_only"`, `"microbiome_only"`, and `"joint"`. Default: a colour-blind-friendly palette.
`show_diagonal`	Logical. If `TRUE`, overlay the chance-level diagonal (AUC = 0.5 reference line). Default: `TRUE`.
`linewidth`	A `numeric(1)` line width for ROC curves. Default: `0.9`.

Value

A `ggplot2` object.

See Also

`MultiOmicsBridgeAnalysis`, `diagnosticClassifier`, `performance`

Examples

```
set.seed(42)
host_counts <- matrix(rpois(500 * 20, 100), nrow = 500, ncol = 20,
    dimnames = list(paste0("Gene", 1:500), paste0("S", 1:20)))
host_counts[1:20, 11:20] <- host_counts[1:20, 11:20] * 5L
mb_counts <- matrix(rpois(60 * 20, 40), nrow = 60, ncol = 20,
    dimnames = list(paste0("Taxon", 1:60), paste0("S", 1:20)))

host_se <- loadHostData(host_counts)
mb_se   <- loadMicrobiomeData(mb_counts)
mae     <- matchSamples(host_se, mb_se)
outcome <- rep(c("ctrl", "treat"), each = 10)
result  <- MultiOmicsBridgeAnalysis(mae, outcome,
                                    n_components = 2,
                                    n_features_host = 20,
                                    n_features_mb = 10,
                                    n_biomarkers = 15, cv_folds = 3)
plotClassifierComparison(result, type = "bar")
```

---

plotIntegration *Joint Biplot of Multi-Omics Integration Results*

Description

Produces a scatter plot of integrated sample scores in the space defined by two DIABLO latent components, with samples coloured by outcome group and optional feature loading vectors overlaid as arrows. This biplot makes it immediately clear how well the multi-omics integration separates the outcome groups and which features drive that separation.

Usage

```
plotIntegration(
  result,
  comp = c(1L, 2L),
  outcome = NULL,
  show_loadings = TRUE,
  n_loading_arrows = 5L,
  point_size = 2.5,
  point_alpha = 0.8,
  colours = NULL
)
```

Arguments

result	A MOBResult object from MultiOmicsBridgeAnalysis.
comp	A length-2 integer vector specifying which two components to plot. Default: c(1, 2).

outcome	A character or factor vector of outcome labels (one per sample), used for point colouring. If NULL, points are plotted in a single colour.
show_loadings	Logical. If TRUE, overlay the top feature loading vectors as arrows. Default: TRUE.
n_loading_arrows	An integer(1) number of top loading arrows to display per omics layer. Default: 5.
point_size	A numeric(1) point size. Default: 2.5.
point_alpha	A numeric(1) point transparency. Default: 0.8.
colours	A named character vector of colours for the outcome groups. If NULL, a default palette is used.

Value

A ggplot2 object.

See Also

MultiOmicsBridgeAnalysis, MOBResult, plotClassifierComparison

Examples

```
library(S4Vectors)
scores <- matrix(rnorm(40), nrow = 20, ncol = 2,
    dimnames = list(paste0("S", 1:20), c("Comp1","Comp2")))
fl <- list(
    host = matrix(rnorm(20), nrow = 10, ncol = 2,
        dimnames = list(paste0("G", 1:10), c("Comp1","Comp2"))),
    microbiome = matrix(rnorm(10), nrow = 5, ncol = 2,
        dimnames = list(paste0("T", 1:5), c("Comp1","Comp2")))
)
bm <- DataFrame(feature = c("G1","T1"), omics_layer = c("host","microbiome"),
                loading_score = c(0.8,0.6), rank = c(1L,2L), component = c(1L,1L))
obj <- MOBResult(scores, fl, bm, list(),
                 params = list(outcome_levels = c("ctrl","treat")))
outcome <- rep(c("ctrl","treat"), each = 10)
plotIntegration(obj, outcome = outcome)
```

---

plotSankey *Feature Flow Diagram: Omics Layer to Biomarkers to Outcome*

Description

Produces a Sankey-style flow diagram showing the pipeline from data source (host or microbiome) through selected top biomarkers to predicted outcome classes. The width of each connection is proportional to the feature's loading score, making it easy to see which features contribute most to separating the outcome groups. The diagram uses base ggplot2 geometry (no additional Sankey packages required).

Usage

```
plotSankey(result, n_features = 10L, colours = NULL, node_width = 0.15)
```

Arguments

result	A MOBResult object from MultiOmicsBridgeAnalysis.
n_features	An integer(1) number of top features to display per omics layer. Default: 10.
colours	A named character vector with colours for "host", "microbiome", and each outcome level. If NULL, defaults are used.
node_width	A numeric(1) width of the node rectangles. Default: 0.15.

Value

A ggplot2 object.

See Also

MultiOmicsBridgeAnalysis, plotIntegration, plotBiomarkerNetwork

Examples

```
library(S4Vectors)
set.seed(42)
scores <- matrix(rnorm(20), nrow = 10, ncol = 2,
    dimnames = list(paste0("S", 1:10), c("Comp1","Comp2")))
fl <- list(
    host = matrix(abs(rnorm(20)), nrow = 10, ncol = 2,
        dimnames = list(paste0("Gene", 1:10), c("Comp1","Comp2"))),
    microbiome = matrix(abs(rnorm(10)), nrow = 5, ncol = 2,
        dimnames = list(paste0("Taxon", 1:5), c("Comp1","Comp2")))
)
bm <- DataFrame(
    feature = c(paste0("Gene", 1:5), paste0("Taxon", 1:3)),
    omics_layer = c(rep("host", 5), rep("microbiome", 3)),
    loading_score = c(0.9, 0.7, 0.5, 0.4, 0.3, 0.8, 0.6, 0.2),
    rank = 1:8, component = rep(1L, 8)
)
obj <- MOBResult(scores, fl, bm, list(),
    params = list(outcome_levels = c("ctrl","treat")))
plotSankey(obj, n_features = 5)
```

show,MOBResult-method *Show method for MOBResult*

Description

Prints a compact summary of a `MOBResult` object, including the top biomarkers, integration method, and classifier AUC-ROC values.

Usage

```
## S4 method for signature 'MOBResult'
show(object)
```

Arguments

`object`	A `MOBResult` object.

Value

Invisibly returns `object`.

Index

∗ **datasets**
example_multiomics, 9

biomarkerDiscovery, *3*, 4, *4*, *7, 8*, *14*, *22, 23*
biomarkers, 6
biomarkers,MOBResult-method *(*biomarkers*)*, 6

diagnosticClassifier, *3, 4*, 7, *15*, *22, 23*, *26*

example_multiomics, 9

featureLoadings, 10
featureLoadings,MOBResult-method *(*featureLoadings*)*, 10

generateReport, *3*, 11

integrationScores, 12
integrationScores,MOBResult-method *(*integrationScores*)*, 12

jointDimReduction, *3–6*, 13, *15*, *22, 23*

loadHostData, *3*, *10*, 15, *17–19*
loadMicrobiomeData, *3*, *10*, 16, *16*, *18, 19*

matchSamples, *3*, *5*, *7*, *10*, *13*, *16, 17*, 18, *21–23*, *25*
MOBResult, *11*, 19, *21*, *23*, *25–29*
MOBResult-class, 21
MultiAssayExperiment, *9*
MultiOmicsBridge *(*MultiOmicsBridge-package*)*, 3
MultiOmicsBridge-package, 3
MultiOmicsBridgeAnalysis, *4*, *6*, *8*, *10, 11*, *14*, *16, 17*, *19*, 21, *21*, *25–29*

performance, 23, *26*
performance,MOBResult-method *(*performance*)*, 23
plotBiomarkerNetwork, *3*, *6*, 24, *29*
plotClassifierComparison, *3*, *8*, *23*, 26, *28*
plotIntegration, *3*, *14*, *23*, *25*, 27, *29*
plotSankey, *3*, 28

show,MOBResult-method, 30
SummarizedExperiment, *9*