Package: dar (via r-universe)

November 25, 2026

Title Differential Abundance Workflows for Microbiome Data

Version 2.0.0

Date 2026-08-04

Description Provides a recipe-based interface for differential abundance analysis of microbiome data. A shared model specification defines covariates, contrasts, reference levels and taxonomic aggregation, with compatibility checks for each method. Pipeable workflows combine preprocessing, quality control and multiple differential abundance methods. Results can be inspected in a common format that preserves method-specific effect scales, visualized and compared. Recipes can be exported and imported using a versioned, non-executable format. Optional consensus rules summarize agreement across methods for each contrast and effect direction.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(roclets = c(``collate'', ``namespace'', ``rd'', ``roxytest::testthat_roclet'', ``roxyglobals::global_roclet''), markdown = TRUE)

URL https://github.com/MicrobialGenomics-IrsicaixaOrg/dar, https://microbialgenomics-irsicaixaorg.github.io/dar/

BugReports https://github.com/MicrobialGenomics-IrsicaixaOrg/dar/issues

biocViews Software, Sequencing, Microbiome, Metagenomics, MultipleComparison, Normalization

Imports checkmate, cli, crayon, dplyr, generics, ggplot2, glue, jsonlite, magrittr, methods, mia, phyloseq, purrr, reformulas, rlang (>= 0.4.11), scales, stringr, tibble, tidyr, withr

Suggests ALDEx2, ALDEx3 (>= 1.3.1), ANCOMBC, apeglm, ashr, Biobase, corncob, ComplexHeatmap, covr, DESeq2, devtools, furrr, future, gplots, heatmaply, knitr, lefser, limma, maaslin3, microbiome, MicrobiomeStat, radEmu (>= 2.3.2.0), readr, rmarkdown, roxygen2, roxyglobals, roxytest, SummarizedExperiment,

TreeSummarizedExperiment, UpSetR, testthat (>= 3.0.0), GenomeInfoDb

Config/testthat/edition 3

Depends R (>= 4.5.0)

LazyData false

Collate 'recipe-class.R' 'aldex2.R' 'aldex3.R' 'analysis-data.R' 'ancom.R' 'bake.R' 'corncob.R' 'dar-package.R' 'data.R' 'deseq2.R' 'effect-plot.R' 'engine-args.R' 'filter_by_abundance.R' 'filter_by_prevalence.R' 'filter_by_rarity.R' 'filter_by_variance.R' 'filter_taxa.R' 'globals.R' 'lefse.R' 'linda.R' 'maaslin3.R' 'misc.R' 'model-adapters.R' 'model.R' 'pkg_check.R' 'plot_methods.R' 'rademu.R' 'rarefaction.R' 'recipe-utils.R' 'recipe_qc.R' 'result-contract.R' 'serialization.R' 'steps_and_checks.R' 'subset_taxa.R' 'utilis-validation.R' 'utils-pipe.R' 'utils-tidy-eval.R' 'utils.R' 'wilcox.R'

VignetteBuilder knitr

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique TRUE

Config/testthat/parallel true

Config/roxygen2/version 8.1.1

Config/pak/sysreqs
cmake libglpk-dev make libgsl0-dev libicu-dev libuv1-dev libxml2-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:51:50 UTC

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

RemoteRef RELEASE_3_24

RemoteSha b625e2bfde2519adbd26c0835b7fd196f69b0c9d

Contents

abundance_plt 4
add_model 5
bake 7
contains_rarefaction 8
cool 8
corr_heatmap 9
effect_plt 10
exclusion_plt 11
export_steps 12
find_intersections 13
get_execution 14
get_model 14

get_phy . . . 15
import_steps . . . 16
intersection_df . . . 17
intersection_plt . . . 17
metaHIV_phy . . . 18
mutual_plt . . . 19
otu_table . . . 20
overlap_df . . . 21
phyloseq_or_null-class . . . 22
prep . . . 22
PrepRecipe-class . . . 23
rand_id . . . 24
rarefaction_help . . . 24
recipe . . . 25
recipe_qc . . . 26
sample_data . . . 27
step_aldex2 . . . 28
step_aldex3 . . . 29
step_ancom . . . 31
step_corncob . . . 34
step_deseq . . . 36
step_filter_by_abundance . . . 39
step_filter_by_prevalence . . . 40
step_filter_by_rarity . . . 41
step_filter_by_variance . . . 42
step_filter_taxa . . . 43
step_lefse . . . 45
step_linda . . . 46
step_maaslin . . . 48
step_rademu . . . 50
step_rarefaction . . . 52
step_subset_taxa . . . 53
step_wilcox . . . 54
steps_ids . . . 55
tax_table . . . 56
test_prep_rec . . . 56
test_rec . . . 57
tidy_abundance . . . 57
tidy_results . . . 58

Index **60**

abundance_plt *Plot observed abundances of selected analysis taxa*

Description

Plot observed abundances of selected analysis taxa

Usage

```
abundance_plt(
  rec,
  taxa_ids = NULL,
  type = "boxplot",
  transform = "compositional",
  scale = 1,
  top_n = 20,
  target = NULL,
  contrast_id = NULL,
  steps = steps_ids(rec, "da"),
  selection = c("shared", "union")
)
```

Arguments

rec	A prepped Recipe object.
taxa_ids	Character vector of analysis taxon IDs to plot. Explicit IDs bypass significance-based selection and are not truncated by top_n.
type	Character vector indicating the type of the result. Options: c("boxplot", "heatmap").
transform	Transformation to apply. The options include: 'compositional' (ie relative abundance), 'Z', 'log10', 'log10p', 'hellinger', 'identity', 'clr', 'alr', or any method from the vegan::decostand function. If the value is NULL, no normalization is applied and works with the raw counts.
scale	Scaling constant for the abundance values when transform = "scale".
top_n	Maximum number of automatically selected taxa. Default: 20.
target	Modeled target used for annotations. Required when a modeled recipe contains more than one target.
contrast_id	Modeled contrast used for automatic taxon selection. Required when taxa_ids = NULL and more than one contrast is available.
steps	Character vector of executed DA step IDs used for automatic selection. Use a single ID to plot that configured method's discoveries. Defaults to all executed DA steps.
selection	Automatic selection rule: "shared" requires a significant result with the same contrast and effect direction in every selected step; "union" includes discoveries from any selected step. No fallback to a different rule is performed when the requested selection is empty.

Details

Abundances use the shared taxonomic analysis view, before any method-specific filtering or transformation. The plotted values are observed abundances, not covariate-adjusted predictions. Transformations use all analysis taxa before the display is restricted to selected IDs. Automatic selections are ordered by decreasing support across selected steps, with ties resolved by analysis taxon order, not by pooled effects.

Value

ggplot2 or HeatmapList, or invisible `NULL` with an informational message when automatic selection finds no significant taxa.

Examples

```
data(test_prep_rec)
results <- tidy_results(test_prep_rec)
contrast_id <- results$contrast_id[[1]]

## Boxplots of taxa detected by all methods
abundance_plt(test_prep_rec, contrast_id = contrast_id)

## Display the same abundances as a heatmap
if (requireNamespace("ComplexHeatmap", quietly = TRUE) &&
    requireNamespace("gplots", quietly = TRUE)) {
  abundance_plt(test_prep_rec, type = "heatmap", contrast_id = contrast_id)
}

## Select taxa explicitly
taxa_ids <- head(unique(results$taxa_id), 3)
abundance_plt(test_prep_rec, taxa_ids = taxa_ids)

## Discoveries from one configured method, without requiring agreement
method_id <- steps_ids(test_prep_rec, "da")[[1]]
abundance_plt(test_prep_rec, steps = method_id, contrast_id = contrast_id)

## Union of discoveries from the executed methods
abundance_plt(test_prep_rec, selection = "union", contrast_id = contrast_id)
```

`add_model` *Add a centralized statistical model to a recipe*

Description

The model is the single source of truth for statistical design arguments. `targets` are the variables of inference; other fixed terms are adjustment variables. `tax_level` defines the taxonomic resolution used by DA methods. When `time` is supplied, numeric values are treated as ordered categorical time points while compiling the model.

Usage

```
add_model(
  rec,
  formula,
  targets = NULL,
  tax_level = NULL,
  time = NULL,
  reference = NULL,
  contrasts = c("simple", "all"),
  na_action = c("error", "drop"),
  max_time_levels = 12
)
```

Arguments

rec	An unprepared Recipe() object.
formula	A one-sided formula or NULL to remove the model.
targets	One or more categorical metadata columns that define the statistical targets. Every target must occur in the fixed formula.
tax_level	A single taxonomic rank used by differential-abundance methods.
time	Optional single metadata column identifying time.
reference	Optional named character vector defining factor reference levels.
contrasts	Either "simple" for pairwise/simple effects or "all" to additionally request difference-in-differences contrasts.
na_action	Either "error" or "drop". Dropping is performed once for all engines during prep().
max_time_levels	Maximum number of time points that may be created by automatic factorization.

Details

DA results, consensus tables and abundance plots share the same aggregated taxonomic units. Taxa with identical lineage through tax_level are summed for analysis; public accessors retain the complete recipe data.

Value

The updated Recipe.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(
    ~ RiskGroup2 + Cluster,
    targets = "RiskGroup2",
    tax_level = "Species"
```

```
  )
get_model(rec)
```

---

bake *Define consensus strategies from a Recipe*

Description

For a prep Recipe adds a consensus strategy to use for result extraction.

Usage

```
bake(
  rec,
  count_cutoff = NULL,
  weights = NULL,
  exclude = NULL,
  id = new_step_id(rec, "bake")
)
```

Arguments

rec	A PrepRecipe object. The step will be added to the sequence of operations for this Recipe.
count_cutoff	Indicates the minimum number of methods in which an OTU must be present (Default: NULL). If count_cutoff is NULL, it is set to length(steps_ids(rec, "da")) - length(exclude).
weights	Named numeric vector with the weight for each method.
exclude	Method ids to exclude.
id	A character string that is unique to this step to identify it.

Value

An object of class PrepRecipe

Examples

```
data(test_prep_rec)

## Require agreement among all differential-abundance methods
cool(bake(test_prep_rec))

## Retain taxa detected by at least one method
cool(bake(test_prep_rec, count_cutoff = 1))

## Exclude a method from the consensus
method_ids <- steps_ids(test_prep_rec, "da")
```

```
cool(bake(test_prep_rec, exclude = method_ids[1]))

## Give the first method twice the weight of the others
weights <- setNames(rep(1, length(method_ids)), method_ids)
weights[1] <- 2
cool(bake(test_prep_rec, weights = weights))
```

`contains_rarefaction` *Checks if Recipe contains a rarefaction step*

Description

Checks if Recipe contains a rarefaction step

Usage

```
contains_rarefaction(rec)
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this recipe.

Value

boolean

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy)
contains_rarefaction(rec)
contains_rarefaction(step_rarefaction(rec))
```

`cool` *Extract results from defined bake*

Description

Extract results from defined bake

Usage

```
cool(rec, bake = 1)
```

Arguments

rec	A PrepRecipe object.
bake	Name or index of the bake to extract.

Value

tbl_df

Examples

```
data(test_prep_rec)
rec <- bake(test_prep_rec)
cool(rec)
```

---

corr_heatmap *Plot otuput of the* overlap_df *function as a heatmap.*

Description

Plot otuput of the overlap_df function as a heatmap.

Usage

```
corr_heatmap(
  rec,
  steps = steps_ids(rec, "da"),
  font_size = 15,
  type = "all",
  target = NULL,
  contrast_id = NULL
)
```

Arguments

rec	A prepped Recipe object.
steps	Character vector with step_ids to take in account.
font_size	Size of the axis font.
type	Indicates whether to use all taxa ("all") or only those that are differentially abundant in at least one method ("da"). Default as "all".
target	Optional modeled target to include.
contrast_id	Optional modeled contrast identifier to include.

Value

heatmap

Examples

```
data(test_prep_rec)

## Correlation heatmap of discoveries shared between methods
if (requireNamespace("heatmaply", quietly = TRUE)) {
  corr_heatmap(test_prep_rec)
}
```

`effect_plt` *Plot method-specific differential-abundance effects*

Description

Displays point estimates returned by the selected differential-abundance steps for one contrast. Each configured step and effect metric has its own panel and axis scales; effects are not pooled or standardized across methods. Colours retain the originating step's significance decision rather than recomputing it from a common p-value threshold. Confidence intervals are not inferred from the common result contract.

Usage

```
effect_plt(
  rec,
  taxa_ids = NULL,
  steps = steps_ids(rec, "da"),
  target = NULL,
  contrast_id = NULL,
  top_n = 20
)
```

Arguments

`rec`	A PrepRecipe object.
`taxa_ids`	Optional character vector of taxon identifiers to display. When supplied, all finite returned effects for these identifiers are included and `top_n` is not applied. Taxa with identical names remain separate units and are labelled with their identifiers when needed.
`steps`	Character vector of executed differential-abundance step IDs. By default, all executed DA steps are included.
`target`	Optional modeled target to include.
`contrast_id`	Optional contrast identifier. This is required when more than one contrast remains after selecting `target`.
`top_n`	Maximum number of taxa selected independently in each step and effect-metric panel when `taxa_ids` is `NULL`. Taxa are ranked by decreasing absolute finite effect within that panel, with taxon ID as a tie-breaker. This is a display selection, not a significance or consensus rule.

Value

An editable ggplot2 object. Returns invisible NULL with an informational message when no finite effects are available. Returned nonfinite effects are omitted; absent result rows are never interpreted as zero effects or nonsignificant findings.

Examples

```
data(test_prep_rec)
results <- tidy_results(test_prep_rec)
effect_plt(
  test_prep_rec,
  steps = results$step_id[[1]],
  contrast_id = results$contrast_id[[1]],
  top_n = 10
)
```

---

exclusion_plt *Plot the number of shared DA OTUs between methods.*

Description

Plot the number of shared DA OTUs between methods.

Usage

```
exclusion_plt(
  rec,
  steps = steps_ids(rec, "da"),
  target = NULL,
  contrast_id = NULL
)
```

Arguments

rec	A prepped Recipe object.
steps	Character vector with step_ids to take in account.
target	Optional modeled target to include.
contrast_id	Optional modeled contrast identifier to include.

Value

ggplot2-class object

Examples

```
data(test_prep_rec)

exclusion_plt(test_prep_rec)

## Exclude the first method from the plot
exclusion_plt(test_prep_rec, steps = steps_ids(test_prep_rec, "da")[-1])
```

`export_steps` *Export a recipe analysis configuration as versioned JSON*

Description

Writes the centralized model, ordered preprocessing and DA steps, and any bake configurations using the non-executable `dar-recipe` JSON schema.

Usage

```
export_steps(rec, file_name)
```

Arguments

`rec`	A `Recipe()` or `PrepRecipe()` object.
`file_name`	Output path.

Value

`file_name`, invisibly.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Species") |>
  step_filter_by_prevalence(threshold = 0.1)
path <- tempfile(fileext = ".json")
export_steps(rec, path)
```

`find_intersections` *Finds common OTU between method results*

Description

Finds common OTU between method results

Usage

```
find_intersections(
  rec,
  steps = steps_ids(rec, "da"),
  target = NULL,
  contrast_id = NULL
)
```

Arguments

`rec`	A Recipe object.
`steps`	character vector with step ids to take in account
`target`	Optional modeled target to include.
`contrast_id`	Optional modeled contrast identifier to include.

Value

tibble

Examples

```
data(test_prep_rec)

## Extract all intersections
find_intersections(test_prep_rec)

## Exclude the first method
find_intersections(
  test_prep_rec,
  steps = steps_ids(test_prep_rec, "da")[-1]
)
```

get_execution *Inspect the recorded execution of a prepared recipe*

Description

`get_execution()` exposes the manifest recorded by `prep()` without recomputing contrasts, modifying results or rerunning any analysis. Samples listed in `dropped_samples` were removed by the centralized model's missing-value policy, not by method-specific filtering.

Usage

```
get_execution(rec)
```

Arguments

`rec`	A PrepRecipe object.

Details

Unrecorded fields are returned as `NULL`.

Value

A list containing the recorded `contrasts`, `dropped_samples`, `executed_steps` and `skipped_steps`. The contrasts and skipped-step tables retain their existing columns, including incompatibility reasons. No centralized model also yields `NULL` contrasts in current recipes. When supplied by an engine, an additional `engine_details` list records its effective settings and package version. ALDEx3 records its seed, scale-model basis, Monte Carlo settings, fitting engine and elapsed time.

Examples

```
data(metaHIV_phy)
prepared <- recipe(metaHIV_phy) |>
  step_filter_by_prevalence(threshold = 0.1) |>
  prep(parallel = FALSE)
get_execution(prepared)
```

get_model *Get the centralized statistical model from a recipe*

Description

Get the centralized statistical model from a recipe

Usage

```
get_model(rec)
```

Arguments

rec	A Recipe() or PrepRecipe() object.

Value

NULL when no model is defined, otherwise a list containing the formula, targets, taxonomic level, and its contrast, reference, missing-value and time controls.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Species")
get_model(rec)
```

---

get_phy *Returns phyloseq from Recipe-class object*

Description

Returns phyloseq from Recipe-class object

Usage

```
get_phy(rec)
```

Arguments

rec	A Recipe or PrepRecipe object.

Value

Phyloseq class object

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy)
get_phy(rec)
```

`import_steps` *Import a recipe configuration from versioned JSON*

Description

Recipe files are treated as untrusted input. The complete document is decoded and validated before supported constructors are applied. Importing bake configurations into an unprepared recipe never runs analysis unless `prepare = TRUE` is supplied explicitly.

Usage

```
import_steps(
  rec,
  file,
  prepare = FALSE,
  parallel = TRUE,
  workers = future::availableCores(constraints = "connections-16")
)
```

Arguments

`rec`	A `Recipe()` or `PrepRecipe()` object.
`file`	Input JSON path.
`prepare`	Whether an unprepared recipe may run `prep()` when the file contains bake configurations. Defaults to `FALSE`.
`parallel`	Passed to `prep()` only when `prepare = TRUE`.
`workers`	Passed to `prep()` only when `prepare = TRUE`.

Value

The configured `Recipe` or `PrepRecipe`.

Examples

```
data(metaHIV_phy)
configured <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Species") |>
  step_filter_by_prevalence(threshold = 0.1)
path <- tempfile(fileext = ".json")
export_steps(configured, path)
restored <- import_steps(recipe(metaHIV_phy), path)
```

intersection_df *Returns data.frame with OTU intersection between methods*

Description

Returns data.frame with OTU intersection between methods

Usage

```
intersection_df(
  rec,
  steps = steps_ids(rec, "da"),
  tidy = FALSE,
  target = NULL,
  contrast_id = NULL
)
```

Arguments

rec	A PrepRecipe object.
steps	character vector with step_ids to take in account.
tidy	Boolean indicating if result must be in tidy format.
target	Optional modeled target to include.
contrast_id	Optional modeled contrast identifier to include.

Value

data.frame class object

Examples

```
data(test_prep_rec)
df <- intersection_df(test_prep_rec)
head(df)
```

intersection_plt *Plot results using UpSet plot*

Description

Plot results using UpSet plot

Usage

```
intersection_plt(
  rec,
  steps = steps_ids(rec, "da"),
  ordered_by = c("freq", "degree"),
  font_size = 2,
  target = NULL,
  contrast_id = NULL
)
```

Arguments

rec	A prepped Recipe object.
steps	Character vector with step_ids to take in account.
ordered_by	How the intersections in the matrix should be ordered by. Options include frequency (entered as "freq"), degree, or both in any order.
font_size	Size of the font.
target	Optional modeled target to include.
contrast_id	Optional modeled contrast identifier to include.

Value

UpSet plot, or invisible NULL with an informational message when no significant taxa are available for the selected hypotheses.

Examples

```
data(test_prep_rec)

if (requireNamespace("UpSetR", quietly = TRUE)) {
  ## UpSet plot of shared discoveries
  intersection_plt(test_prep_rec)

  ## Order intersections by the number of methods involved
  intersection_plt(test_prep_rec, ordered_by = "degree")
}
```

---

metaHIV_phy *Phyloseq object from metaHIV project*

Description

A phyloseq object containing the abundance table, sample metadata and taxonomic annotations supplied for the metaHIV project associated with Noguera-Julian et al. (2016). It is provided as a software example. The profiling pipeline and the measurement units of this derived table have not been independently verified; integer values alone do not establish that a table contains raw read counts.

Usage

```
data("metaHIV_phy")
```

Format

A phyloseq object with 451 taxa, 156 samples, 3 sample variables and 7 taxonomic ranks.

Source

Noguera-Julian et al. (2016), doi:10.1016/j.ebiom.2016.01.032.

---

mutual_plt *Mutual finding plot*

Description

Plots number of differentially abundant features mutually found by defined number of methods, colored by the differential abundance direction and separated by comparison.

Usage

```
mutual_plt(
  rec,
  count_cutoff = NULL,
  comparisons = NULL,
  steps = steps_ids(rec, type = "da"),
  top_n = 20,
  target = NULL,
  contrast_id = NULL
)
```

Arguments

rec	A prepped Recipe object.
count_cutoff	Indicates the minimum number of methods in which an OTU must be present (Default: NULL). If count_cutoff is NULL count_cutoff is equal to `length(steps_ids(rec, "da")) * 2 / 3`.
comparisons	By default, this function plots all comparisons. However, if the user indicates the comparison or comparisons of interest, only the selected ones will be plotted.
steps	Character vector with step_ids to take in account. Default all "da" methods.
top_n	Maximum number of taxa to represent. Default: 20.
target	Optional modeled target to include.
contrast_id	Optional modeled contrast identifier to include.

Value

ggplot2, or invisible NULL with an informational message when no significant taxa are available for the selected hypotheses.

Examples

```
data(test_prep_rec)

mutual_plt(test_prep_rec)

## Require agreement among all methods
mutual_plt(
  test_prep_rec,
  count_cutoff = length(steps_ids(test_prep_rec, "da"))
)

## Show a single comparison
comparison <- tidy_results(test_prep_rec)$comparison[[1]]
mutual_plt(test_prep_rec, comparisons = comparison)

## Exclude the first method from the plot
mutual_plt(test_prep_rec, steps = steps_ids(test_prep_rec, "da")[-1])
```

---

otu_table *Extract a canonical count table from a recipe*

Description

Extract a canonical count table from a recipe

Usage

```
otu_table(rec)
```

Arguments

rec	A Recipe or PrepRecipe object.

Value

A wide tibble with one row per taxon, taxa_id first, and one column per sample. Taxa and samples follow phyloseq::taxa_names() and phyloseq::sample_names() regardless of the stored OTU-table orientation.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy)
otu_table(rec)
```

overlap_df *Overlap of significant OTUs between tested methods.*

Description

Overlap of significant OTUs between tested methods.

Usage

```
overlap_df(
  rec,
  steps = steps_ids(rec, "da"),
  type = "all",
  target = NULL,
  contrast_id = NULL
)
```

Arguments

rec	A PrepRecipe object.
steps	Character vector with step_ids to take in account.
type	Indicates whether to use all taxa ("all") or only those that are differentially abundant in at least one method ("da"). Default as "all".
target	Optional modeled target to include.
contrast_id	Optional modeled contrast identifier to include.

Value

df

Examples

```
data(test_prep_rec)
df <- overlap_df(test_prep_rec)
head(df)
```

phyloseq_or_null-class *Recipe-class object*

Description

A Recipe is a description of the steps to be applied to a data set in order to prepare it for data analysis.

Usage

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

Arguments

object	A Recipe object.

Value

Recipe-class object

Slots

`phyloseq` Phyloseq-class object.

`steps` List of step-class objects that will be used by DA.

`model` Optional centralized statistical model specification created by `add_model()`.

prep *Performs all the steps defined in a Recipe*

Description

For a Recipe with at least one preprocessing or DA operation run the steps in a convenient order.

Usage

```
prep(
  rec,
  parallel = TRUE,
  workers = future::availableCores(constraints = "connections-16"),
  force = FALSE
)
```

Arguments

rec	A Recipe object.
parallel	if FALSE, no palatalization. if TRUE, parallel execution using future and furrr packages.
workers	Number of workers for palatalization.
force	Force the reexecution of all steps. This remove previous results.

Details

Recipes containing differential-abundance steps require an explicit add_model() before execution. Preprocessing-only recipes remain supported without a model. A missing model is reported before preprocessing or dependency checks, so the input data are not modified.

Value

A PrepRecipe object.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Class") |>
  step_filter_by_prevalence(threshold = 0.03) |>
  step_wilcox()

da_results <- prep(rec, parallel = FALSE)
```

PrepRecipe-class *PrepRecipe-class object*

Description

A PrepRecipe is Recipe with the results corresponding to the steps defined in the Recipe.

Value

PrepRecipe-class object

Slots

results Contains the results of all defined analysis in the Recipe.

bakes Contains the executed bakes.

execution Manifest containing resolved contrasts, samples removed by the missing-value policy, and executed or skipped methods.

rand_id *Make a random identification field for steps*

Description

Make a random identification field for steps

Usage

```
rand_id(prefix = "step")
```

Arguments

prefix	A single character string

Value

A character string with the prefix and random letters separated by and underscore.

Examples

```
rand_id("step")
```

rarefaction_help *Information about the Rarefaction process*

Description

Prints detailed information regarding the use of rarefaction in microbiome analysis, its implications, and relevant literature.

Usage

```
rarefaction_help()
```

Value

invisible

Examples

```
rarefaction_help()
```

recipe *Create a Recipe for preprocessing data*

Description

A Recipe is a description of the steps to be applied to a data set in order to prepare it for data analysis.

Usage

```
recipe(microbiome_object = NULL, steps = list())
```

Arguments

microbiome_object	Phyloseq-class object or TreeSummarizedExperiment-class object.
steps	list with steps.

Value

An object of class Recipe containing the microbiome object, an optional centralized model and the configured processing or DA steps.

Examples

```
data(metaHIV_phy)

## Configure, prepare and obtain a consensus
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum") |>
  step_filter_by_prevalence(threshold = 0.3) |>
  step_deseq() |>
  step_maaslin()

da_results <- prep(rec, parallel = FALSE) |>
  bake()
cool(da_results)

## A recipe without a model can be used for preprocessing
rec <- recipe(metaHIV_phy) |>
  step_filter_by_prevalence(threshold = 0.1)
```

recipe_qc *Recipe quality-control metrics*

Description

recipe_qc() summarizes sparsity and sequencing-depth metrics from the count matrix stored in a Recipe or PrepRecipe object. The summary is independent of the centralized statistical model and can therefore be used before model configuration and after recipe preprocessing.

Usage

```
recipe_qc(rec, group_by = NULL)
```

Arguments

rec	A Recipe or PrepRecipe object.
group_by	NULL or the name of one sample metadata column used to stratify the QC summary.

Details

With group_by = NULL, the result contains one overall row. When a sample metadata column is supplied, the overall row is followed by one row for each observed group. The group_by and group columns are NA in the overall row, so a real group named "all" remains unambiguous.

The returned metrics are:

- n: number of values in the count matrix.
- n_zero: number of zero values.
- pct_zero: percentage of zero values.
- pct_all_zero: percentage of taxa absent from every sample.
- pct_singletons: percentage of taxa detected in exactly one sample.
- pct_doubletons: percentage of taxa detected in exactly two samples.
- n_samples: number of samples.
- lib_size_min: minimum sample library size.
- lib_size_max: maximum sample library size.
- count_mean: mean of the per-sample mean counts.
- count_max: mean of the per-sample maximum counts.

Value

A tibble with grouping identifiers and QC metrics.

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy)

## Overall metrics do not require a statistical model
recipe_qc(rec)

## Add explicit sample-level stratification when useful
recipe_qc(rec, group_by = "RiskGroup2")
```

`sample_data` *Extract complete sample metadata from a recipe*

Description

Extract complete sample metadata from a recipe

Usage

```
sample_data(rec)
```

Arguments

`rec`	A `Recipe` or `PrepRecipe` object.

Value

A tibble with `sample_id` followed by every sample metadata column. Samples are ordered according to `phyloseq::sample_names()`. If the recipe has no sample metadata, an ID-only tibble is returned.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy)
sample_data(rec)
```

`step_aldex2` *ALDEx2 analysis*

Description

Differential abundance analysis using ALDEx2's Monte Carlo log-ratio transformation and native statistical tests. Requires `add_model()`, fits a GLM for each planned contrast and reports its coefficient and adjusted P value.

Usage

```
step_aldex2(
  rec,
  max_significance = 0.05,
  mc.samples = 128,
  denom = "all",
  rarefy = FALSE,
  id = new_step_id(rec, "aldex2"),
  engine_args = list()
)
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires `add_model()` before `prep()`; the step can be configured before adding the model.
`max_significance`	Adjusted P value cutoff for the GLM.
`mc.samples`	The number of Monte Carlo instances to use to estimate the underlying distributions. The default is 128.
`denom`	Features used as the geometric-mean denominator: `"all"` (default) or explicit numeric row indices in the aggregated count matrix. Choose indices a priori and account for native removal of all-zero taxa. Data-dependent denominators are not supported with the shared design.
`rarefy`	Boolean indicating if OTU counts must be rarefyed. This rarefaction uses the standard R sample function to resample from the abundance values in the otu_table component of the first argument, physeq. Often one of the major goals of this procedure is to achieve parity in total number of counts between samples, as an alternative to other formal normalization procedures, which is why a single value for the sample.size is expected. If 'no_seed', rarefaction is performed without a set seed.
`id`	Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names `taxa_id`, `contrast_id`, `comparison`, `contrast_type`, `var`, `effect`, `sum`, `.tax_order`, `.contrast_order` or `.effect_order`. Other identifiers, including names such as `.custom`, are retained unchanged.

engine_args — Named lists of advanced arguments for the native clr and fit stages. Effects come from the GLM. Arguments managed by dar or exposed above cannot be overridden.

Details

With add_model(), effect is the contrast-oriented coefficient from ALDEx2::aldex.glm(), averaged over Monte Carlo instances, and effect_metric is "aldex_glm_coefficient". It comes from the same GLM as the reported adjusted P value. Under the default denominator, scale and Gaussian identity-link settings, the response is CLR in base 2; this coefficient is not an absolute-abundance log2 fold change or the standardized effect from ALDEx2::aldex.effect(). Advanced native settings can change the response or link scale.

Value

An object of class Recipe

See Also

Other Diff taxa steps: step_aldex3(), step_ancom(), step_corncob(), step_deseq(), step_lefse(), step_linda(), step_maaslin(), step_rademu(), step_wilcox()

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum")

rec_default <- step_aldex2(rec)

# Increase the number of Monte Carlo instances.
rec_more_instances <- step_aldex2(rec, mc.samples = 256)
```

---

step_aldex3 *ALDEx3 fixed- and mixed-effects analysis*

Description

Fit planned contrasts with the official ALDEx3 engine. Fixed models use native linear regression; mixed models use exact lme4 fits, not the approximate blmm engine. A centralized add_model() is required.

Usage

```
step_aldex3(
  rec,
  max_significance = 0.05,
  nsample = 2000,
  scale_model = "clr",
  scale_args = list(gamma = 0.5),
  p_adj_method = "BH",
  n_cores = 1,
  seed = 130L,
  rarefy = FALSE,
  id = new_step_id(rec, "aldex3"),
  engine_args = list()
)
```

Arguments

rec	An unprepared Recipe object.
max_significance	Native adjusted P value cutoff.
nsample	Number of Monte Carlo draws passed to ALDEx3. Default 2000.
scale_model	One of "clr", "tss", "sample", "coefficient".
scale_args	Named list of scale-model parameters; see Details.
p_adj_method	Native multiple-testing adjustment, default "BH".
n_cores	Number of cores for native mixed-model fitting, default one.
seed	Integer Monte Carlo seed. Restores the caller's RNG state.
rarefy	Whether to rarefy before fitting; default FALSE.
id	Unique step identifier.
engine_args	Named list with an optional fit list. Supports native test ("t.HC3", "t.HC0", "t") for fixed models and streamsize. Validation is left to ALDEx3. The native test setting does not apply to mixed models. Data, model, scale, engine, seed and two-sided inference are managed by dar.

Details

Supply sequence counts for inference. Taxonomic aggregation and common preprocessing precede native Dirichlet sampling. No extra prevalence filter or implicit "other" feature is added. Filtering changes the composition and the features defining CLR scale assumptions. External scale information must describe the retained community; removed counts are not reconstructed. Native numerical parameters and input values are passed to ALDEx3. Engine-specific validation and errors are left to that package. dar checks its configuration and taxon, sample and contrast identities.

Scale models are specified by name, not executable functions. "clr" and "tss" accept scale_args = list(gamma = 0.5), the native default SD on the log2 coefficient scale. gamma = 0 fixes the corresponding scale assumption. "sample" requires named s.mu and s.sd vectors of external log2-scale means and SDs, one per input sample ID. "coefficient" requires named c.mu and c.sd vectors,

one per column of the resolved fixed-effects design. This interface supports independent SDs, not covariance matrices or arbitrary scale functions. The default is a modeling choice, not an estimate of scale uncertainty. Coefficient coding and the units of numeric covariates affect these priors; choose scale information appropriate to the study.

Scale uncertainty and priors always refer to the original resolved design, even when a planned contrast requires releveling. Each contrast uses the same seed and scale-model basis. The native model is refitted for each contrast; mixed models can therefore be expensive.

Effects are mean Monte Carlo coefficients on the log2 abundance scale conditional on the specified scale model (`aldex3_log2_abundance_coefficient`). Sequence counts alone do not measure absolute microbial load. P values and adjusted P values retain ALDEx3's native directional Monte Carlo procedure, including adjustment across taxa separately for each coefficient. They are not BH applied to averaged P values, nor a joint adjustment over all planned contrasts. Missing statistics remain missing. Execution settings and the effective package version are recorded by `get_execution()`.

Value

A Recipe with the ALDEx3 step added.

See Also

Other Diff taxa steps: `step_aldex2()`, `step_ancom()`, `step_corncob()`, `step_deseq()`, `step_lefse()`, `step_linda()`, `step_maaslin()`, `step_rademu()`, `step_wilcox()`

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Genus") |>
  step_aldex3(scale_model = "clr", scale_args = list(gamma = 0.5))
# Use verified sequence counts for inference; prep() executes the recipe.
```

---

step_ancom *ANCOM analysis*

Description

Determine taxa whose absolute abundances, per unit volume, of the ecosystem (e.g., gut) are significantly different with changes in the covariate of interest (e.g., group). The current version of ancombc2 function implements Analysis of Compositions of Microbiomes with Bias Correction (ANCOM-BC2) in cross-sectional and repeated measurements data. In addition to the two-group comparison, ANCOM-BC2 also supports testing for continuous covariates and multi-group comparisons. This adapter executes the planned contrasts from `add_model()`, not the native global, Dunnett or trend tests.

Usage

```
step_ancom(
  rec,
  fix_formula = NULL,
  rand_formula = NULL,
  p_adj_method = "holm",
  prv_cut = 0.1,
  lib_cut = 0,
  s0_perc = 0.05,
  group = NULL,
  struc_zero = FALSE,
  neg_lb = FALSE,
  alpha = 0.05,
  n_cl = 1,
  verbose = FALSE,
  global = FALSE,
  pairwise = FALSE,
  dunnet = FALSE,
  trend = FALSE,
  rarefy = FALSE,
  id = new_step_id(rec, "ancom"),
  engine_args = list()
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires add_model() before prep(); the step can be configured before adding the model.
fix_formula	the character string expresses how the microbial absolute abundances for each taxon depend on the fixed effects in metadata. When specifying the fix_formula, make sure to include the group variable in the formula if it is not NULL. Defaults to NULL: use add_model(). An explicit formula is overridden with a warning by the shared model.
rand_formula	the character string expresses how the microbial absolute abundances for each taxon depend on the random effects in metadata. ANCOM-BC2 follows the lmerTest package in formulating the random effects. See ?lmerTest::lmer for more details. Default is NULL.
p_adj_method	character. method to adjust p-values. Default is "holm". Options include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". See ?stats::p.adjust for more details.
prv_cut	a numerical fraction between 0 and 1. Taxa with prevalences less than prv_cut will be excluded in the analysis. For instance, suppose there are 100 samples, if a taxon has nonzero counts presented in less than 10 samples, it will not be further analyzed. Default is 0.10.
lib_cut	a numerical threshold for filtering samples based on library sizes. Samples with library sizes less than lib_cut will be excluded in the analysis. Default is 0, i.e.

do not discard any sample.

`s0_perc` a numerical fraction between 0 and 1. Inspired by Significance Analysis of Microarrays (SAM) methodology, a small positive constant is added to the denominator of ANCOM-BC2 test statistic corresponding to each taxon to avoid the significance due to extremely small standard errors, especially for rare taxa. This small positive constant is chosen as s0_perc-th percentile of standard error values for each fixed effect. Default is 0.05 (5th percentile).

`group` character. The name of the group variable in metadata. group should be discrete. Specifying group is required for detecting structural zeros and performing multi-group comparisons (global test, pairwise directional test, Dunnett's type of test, and trend test). Default is NULL. If the group of interest contains only two categories, leave it as NULL.

`struc_zero` logical. Whether to detect structural zeros based on group. Default is FALSE. See Details for a more comprehensive discussion on structural zeros.

`neg_lb` logical. Whether to classify a taxon as a structural zero using its asymptotic lower bound. Default is FALSE.

`alpha` numeric. Level of significance. Default is 0.05.

`n_cl` numeric. The number of nodes to be forked. For details, see ?parallel::makeCluster. Default is 1 (no parallel computing).

`verbose` logical. Whether to generate verbose output during the ANCOM-BC2 fitting process. Default is FALSE.

`global` logical. Whether to perform the global test. Default is FALSE.

`pairwise` logical. Whether to perform the pairwise directional test. Default is FALSE.

`dunnet` logical. Whether to perform the Dunnett's type of test. Default is FALSE.

`trend` logical. Whether to perform trend test. Default is FALSE.

`rarefy` Boolean indicating if OTU counts must be rarefyed. This rarefaction uses the standard R sample function to resample from the abundance values in the otu_table component of the first argument, physeq. Often one of the major goals of this procedure is to achieve parity in total number of counts between samples, as an alternative to other formal normalization procedures, which is why a single value for the sample.size is expected. If 'no_seed', rarefaction is performed without a set seed.

`id` Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names `taxa_id`, `contrast_id`, `comparison`, `contrast_type`, `var`, `effect`, `sum`, `.tax_order`, `.contrast_order` or `.effect_order`. Other identifiers, including names such as `.custom`, are retained unchanged.

`engine_args` Named lists of advanced arguments for the native `fit` stage. Arguments managed by dar or exposed above cannot be overridden.

Details

Significance follows ANCOM-BC2's sensitivity-aware `diff_robust` decision when `pseudo_sens = TRUE` (the default), rather than the basic `diff` decision alone. `engine_args = list(fit = list(pseudo_sens = FALSE))` explicitly disables sensitivity analysis and uses `diff`. Adjusted p-values and effect estimates are not changed by this selection.

Value

An object of class Recipe

See Also

Other Diff taxa steps: step_aldex2(), step_aldex3(), step_corncob(), step_deseq(), step_lefse(), step_linda(), step_maaslin(), step_rademu(), step_wilcox()

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Genus")

rec_default <- step_ancom(rec)

# Pass advanced convergence controls to ANCOM-BC2.
rec_custom <- step_ancom(
  rec,
  engine_args = list(
    fit = list(iter_control = list(tol = 1e-3, max_iter = 50))
  )
)

# Rarefy counts only when this step runs.
rec_rarefied <- step_ancom(rec, rarefy = TRUE)
```

---

step_corncob *corncob analysis*

Description

Corncob an individual taxon regression model that uses abundance tables and sample data. corncob is able to model differential abundance and differential variability, and addresses each of the challenges presented below:

Usage

```
step_corncob(
  rec,
  phi.formula = stats::formula(~1),
  formula_null = NULL,
  phi.formula_null = NULL,
  link = "logit",
  phi.link = "logit",
  test = "Wald",
  boot = FALSE,
```

```
  B = 1000,
  filter_discriminant = TRUE,
  fdr_cutoff = 0.05,
  fdr = "fdr",
  log2FC = 0,
  rarefy = FALSE,
  id = new_step_id(rec, "corncob"),
  engine_args = list()
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires add_model() before prep(); the step can be configured before adding the model.
phi.formula	An object of class formula without the response: a symbolic description of the model to be fitted to the dispersion.
formula_null	Retained design argument, default NULL. Explicit values are overridden with a warning by the add_model() contrasts.
phi.formula_null	Formula for overdispersion under null, without response. Default NULL. Explicit values are overridden with a warning by the add_model() contrasts.
link	Link function for abundance covariates, defaults to "logit".
phi.link	Link function for dispersion covariates, defaults to "logit".
test	Hypothesis test, default "Wald". Only Wald contrasts are supported; other tests make the step incompatible with the shared model.
boot	Whether to request bootstrap inference, default FALSE. Bootstrap is unsupported and makes the step incompatible.
B	Retained bootstrap iteration setting. Not used for Wald contrasts.
filter_discriminant	Boolean. Defaults to TRUE. If FALSE, discriminant taxa will not be filtered out.
fdr_cutoff	Integer. Defaults to 0.05. Desired type 1 error rate.
fdr	Character. Defaults to "fdr". False discovery rate control method, see p.adjust for more options.
log2FC	Minimum absolute effect on the reported coefficient scale. Despite the argument name, a logit-link coefficient is not a log2 fold change. Default zero.
rarefy	Boolean indicating if OTU counts must be rarefyed. This rarefaction uses the standard R sample function to resample from the abundance values in the otu_table component of the first argument, physeq. Often one of the major goals of this procedure is to achieve parity in total number of counts between samples, as an alternative to other formal normalization procedures, which is why a single value for the sample.size is expected. If 'no_seed', rarefaction is performed without a set seed.

id	Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names taxa_id, contrast_id, comparison, contrast_type, var, effect, sum, .tax_order, .contrast_order or .effect_order. Other identifiers, including names such as .custom, are retained unchanged.
engine_args	Named lists of advanced arguments for the native fit stage. Arguments managed by dar or exposed above cannot be overridden.

Details

- different sequencing depth
- excessive zeros from unobserved taxa
- high variability of empirical relative abundances (overdispersion)
- within-taxon correlation
- hypothesis testing with categorical and continuous covariates

Centralized models use non-bootstrap Wald contrasts. With engine_args = list(fit = list(robust = TRUE)), inference uses corncob's native sandwich covariance rather than the inverse Hessian. This does not add support for random effects or repeated-measures dependence.

Value

An object of class Recipe

See Also

Other Diff taxa steps: step_aldex2(), step_aldex3(), step_ancom(), step_deseq(), step_lefse(), step_linda(), step_maaslin(), step_rademu(), step_wilcox()

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum")

rec_default <- step_corncob(rec)

# Use a stricter significance threshold.
rec_strict <- step_corncob(rec, fdr_cutoff = 0.01)
```

---

step_deseq *DESeq2 analysis*

Description

Differential expression analysis based on the Negative Binomial (a.k.a. Gamma-Poisson) distribution. This function performs a default analysis through the steps: 1) estimation of size factors: `estimateSizeFactors`. 2) estimation of dispersion: `estimateDispersions`. 3) Negative Binomial GLM fitting and Wald statistics: nbinomWaldTest. For complete details on each step, see the manual pages of the respective functions. After the DESeq function returns a DESeqDataSet object, results tables (log2 fold changes and p-values) can be generated using the results function. Shrunken LFC can then be generated using the lfcShrink function.

Usage

```
step_deseq(
  rec,
  test = "Wald",
  fitType = "local",
  betaPrior = FALSE,
  type = "ashr",
  max_significance = 0.05,
  log2FC = 0,
  rarefy = FALSE,
  id = new_step_id(rec, "deseq"),
  engine_args = list()
)
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires `add_model()` before `prep()`; the step can be configured before adding the model.
`test`	Either "Wald" or "LRT", which will then use either Wald significance tests (defined by nbinomWaldTest), or the likelihood ratio test on the difference in deviance between a full and reduced model formula (defined by nbinomLRT).
`fitType`	either "parametric", "local", "mean", or "glmGamPoi" for the type of fitting of dispersions to the mean intensity. See estimateDispersions for description.
`betaPrior`	whether or not to put a zero-mean normal prior on the non-intercept coefficients See nbinomWaldTest for description of the calculation of the beta prior. In versions >=1.16, the default is set to FALSE, and shrunken LFCs are obtained afterwards using lfcShrink.
`type`	"apeglm" is the adaptive Student's t prior shrinkage estimator from the 'apeglm' package; "ashr" is the adaptive shrinkage estimator from the 'ashr' package, using a fitted mixture of normals prior - see the Stephens (2016) reference below for citation; "normal" is the 2014 DESeq2 shrinkage estimator using a Normal prior.
`max_significance`	The q-value threshold for significance.
`log2FC`	log2FC cutoff.

rarefy	Boolean indicating if OTU counts must be rarefyed. This rarefaction uses the standard R sample function to resample from the abundance values in the otu_table component of the first argument, physeq. Often one of the major goals of this procedure is to achieve parity in total number of counts between samples, as an alternative to other formal normalization procedures, which is why a single value for the sample.size is expected. If 'no_seed', rarefaction is performed without a set seed.
id	Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names `taxa_id`, `contrast_id`, `comparison`, `contrast_type`, `var`, `effect`, `sum`, `.tax_order`, `.contrast_order` or `.effect_order`. Other identifiers, including names such as `.custom`, are retained unchanged.
engine_args	Named lists of advanced arguments for the native `size_factors`, `fit`, `results`, or `shrink` stage. Arguments managed by dar or exposed above cannot be overridden.

Details

Size factors use DESeq2's native `type = "poscounts"` estimator, which accommodates zeros without adding pseudocounts. Results are extracted with `alpha = max_significance` before effect-size shrinkage. The same results object is passed to `lfcShrink()`, retaining native p-values, adjusted p-values and missingness with the default shrinkage settings. Advanced `results` controls apply both with and without shrinkage.

Value

An object of class `Recipe`

See Also

Other Diff taxa steps: `step_aldex2()`, `step_aldex3()`, `step_ancom()`, `step_corncob()`, `step_lefse()`, `step_linda()`, `step_maaslin()`, `step_rademu()`, `step_wilcox()`

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum")

rec_default <- step_deseq(rec)

# Pass advanced arguments to native DESeq2 stages.
rec_custom <- step_deseq(
  rec,
  engine_args = list(
    fit = list(minReplicatesForReplace = Inf),
    results = list(pAdjustMethod = "BH"),
    shrink = list(lfcThreshold = 1)
  )
)
```

step_filter_by_abundance *Filter taxa by abundance*

Description

This is a convenience wrapper around the filter_taxa function. It is intended to speed up filtering complex experimental objects with one function call. In the case of filter_by_abundance, the filtering will be based on the relative abundance of each taxon. The taxa retained in the dataset are those where the sum of their abundance is greater than the product of the total abundance and the provided threshold.

Usage

```
step_filter_by_abundance(
  rec,
  threshold = 0.01,
  id = new_step_id(rec, "filter_by_abundance")
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe.
threshold	The relative abundance threshold for filtering taxa, expressed as a proportion of the total abundance. For example, a threshold of 0.01 means that a taxon must make up at least 1% of the total abundance to be retained. The default value is 0.01.
id	A character string that is unique to this step to identify it.

Details

The function calculates the total abundance of all taxa in the phyloseq object. It then compares this total abundance to the abundance of each individual taxon. If a taxon's abundance is less than the threshold times the total abundance, that taxon is removed from the phyloseq object.

Value

A Recipe object that has been filtered based on abundance.

Note

This function modifies rec in place, you might want to make a copy of rec before modifying it if you need to preserve the original object.

See Also

filter_taxa

Other filter phy steps: step_filter_by_prevalence(), step_filter_by_rarity(), step_filter_by_variance(), step_filter_taxa()

Examples

```
data(metaHIV_phy)

recipe(metaHIV_phy) |>
  step_filter_by_abundance()
```

---

step_filter_by_prevalence

Filter taxa by prevalence

Description

This is a convenience function around the filter_taxa function. It is designed to speed up filtering complex experimental objects with one function call. In the case of run_filter_by_prevalence, the filtering will be based on the prevalence of each taxon. The taxa retained in the dataset are those where the prevalence is greater than the provided threshold.

Usage

```
step_filter_by_prevalence(
  rec,
  threshold = 0.01,
  id = new_step_id(rec, "filter_by_prevalence")
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe.
threshold	The prevalence threshold for filtering taxa, expressed as a proportion of the total number of samples. For example, a threshold of 0.01 means that a taxon must be present in at least 1% of the samples to be retained. The default value is 0.01.
id	A character string that is unique to this step to identify it.

Details

The function calculates the prevalence of all taxa in the phyloseq object as the proportion of samples in which they are present. It then compares this prevalence to the threshold. If a taxon's prevalence is less than the threshold, that taxon is removed from the phyloseq object.

Value

A Recipe object that has been filtered based on prevalence.

Note

This function modifies `rec` in place, you might want to make a copy of `rec` before modifying it if you need to preserve the original object.

See Also

`filter_taxa`

Other filter phy steps: `step_filter_by_abundance()`, `step_filter_by_rarity()`, `step_filter_by_variance()`, `step_filter_taxa()`

Examples

```
data(metaHIV_phy)

recipe(metaHIV_phy) |>
  step_filter_by_prevalence()
```

---

step_filter_by_rarity *Filter taxa by rarity*

Description

This is a convenience function around the filter_taxa function. It is designed to speed up filtering complex experimental objects with one function call. In the case of run_filter_by_rarity, the filtering will be based on the rarity of each taxon. The taxa retained in the dataset are those where the prevalence (proportion of samples where present) is LESS than the provided threshold.

Usage

```
step_filter_by_rarity(
  rec,
  threshold = 0.01,
  id = new_step_id(rec, "filter_by_rarity")
)
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this Recipe.
`threshold`	The rarity threshold for filtering taxa, expressed as a proportion of the total number of samples. For example, a threshold of 0.01 means that a taxon must be present in less than 1% of the samples to be retained. The default value is 0.01.
`id`	A character string that is unique to this step to identify it.

Details

The function calculates the prevalence of all taxa in the phyloseq object as the proportion of samples in which they are present. It then compares this prevalence to the threshold. If a taxon's prevalence is greater than or equal to the threshold, that taxon is removed (filtered out) from the phyloseq object, leaving only the "rare" taxa.

Value

A Recipe object that has been filtered based on rarity.

Note

This function modifies `rec` in place, you might want to make a copy of `rec` before modifying it if you need to preserve the original object.

See Also

`filter_taxa`

Other filter phy steps: `step_filter_by_abundance()`, `step_filter_by_prevalence()`, `step_filter_by_variance()`, `step_filter_taxa()`

Examples

```
data(metaHIV_phy)

recipe(metaHIV_phy) |>
  step_filter_by_rarity()
```

---

step_filter_by_variance

Filter taxa by variance

Description

This is a convenience function around the filter_taxa function. It is designed to speed up filtering complex experimental objects with one function call. In the case of run_filter_by_variance, the filtering will be based on the variance of each taxon. The taxa retained in the dataset are those where the variance of their abundance is greater than the provided threshold.

Usage

```
step_filter_by_variance(
  rec,
  threshold = 0.01,
  id = new_step_id(rec, "filter_by_variance")
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe.
threshold	The variance threshold for filtering taxa. The default value is 0.01.
id	A character string that is unique to this step to identify it.

Details

The function calculates the variance of all taxa in the phyloseq object. It then compares this variance to the variance of each individual taxon. If a taxon's variance is less than or equal to the threshold, that taxon is removed from the phyloseq object.

Value

A Recipe object that has been filtered based on variance.

Note

This function modifies rec in place, you might want to make a copy of rec before modifying it if you need to preserve the original object.

See Also

filter_taxa

Other filter phy steps: step_filter_by_abundance(), step_filter_by_prevalence(), step_filter_by_rarity(), step_filter_taxa()

Examples

```
data(metaHIV_phy)

recipe(metaHIV_phy) |>
  step_filter_by_variance()
```

---

step_filter_taxa *Filter taxa based on across-sample OTU abundance criteria*

Description

Add a preprocessing step that applies one safely serializable predicate to each taxon's abundance vector across samples. Filtering is performed during prep(); taxa for which the predicate returns FALSE are removed.

Usage

```
step_filter_taxa(rec, .f, id = new_step_id(rec, "filter_taxa"))
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe.
.f	A function with exactly one argument named x, without a default, or a character string containing that function expression. It must return a single logical value for the abundance vector x. Only the restricted expression syntax described below is accepted; lists of functions are not supported.
id	A character string that is unique to this step to identify it.

Details

The predicate body must be a single expression, without braces, assignments, external variables or user-defined helper calls. Write thresholds as literal numbers inside the expression, even when using the function directly in R. These restrictions keep recipes safely exportable and importable; they are checked when the step is added, not just on export.

Accepted expressions use x, finite numeric or non-missing logical literals, parentheses, arithmetic operators (+, -, *, /, ^, %%, %/%), comparisons (>, >=, <, <=, ==, !=), logical operators (&, |, !), and calls to sum, length, mean, median, min, max, var, sd, any, all or is.na. The only supported named argument is na.rm in sum, mean, median, min, max, var, sd, any and all. Namespace-qualified calls are not accepted.

Value

A Recipe with the filtering step added. The constructor does not filter the data or return a logical vector. During prep(), the step uses phyloseq::filter_taxa() with prune = TRUE; there is no prune argument in this wrapper.

See Also

Other filter phy steps: step_filter_by_abundance(), step_filter_by_prevalence(), step_filter_by_rarity(), step_filter_by_variance()

Examples

```
data(metaHIV_phy)

## Keep taxa present in at least 3% of samples
recipe(metaHIV_phy) |>
  step_filter_taxa(.f = function(x) sum(x > 0) >= 0.03 * length(x)) |>
  prep()

## The same predicate can be supplied as a string
recipe(metaHIV_phy) |>
  step_filter_taxa(.f = "function(x) sum(x > 0) >= 0.03 * length(x)")
```

`step_lefse` *lefse analysis*

Description

Lefser is metagenomic biomarker discovery tool that is based on LEfSe tool and is published by Huttenhower et al. 2011. Lefser is the R implementation of the LEfSe method. Using statistical analyses, lefser compares microbial populations of healthy and diseased subjects to discover differencially expressed microorganisms. Lefser than computes effect size, which estimates magnitude of differential expression between the populations for each differentially expressed microorganism. Subclasses of classes can also be assigned and used within the analysis.

Usage

```
step_lefse(
  rec,
  kruskal.threshold = 0.05,
  wilcox.threshold = 0.05,
  lda.threshold = 2,
  subclassCol = NULL,
  assay = 1L,
  trim.names = FALSE,
  rarefy = TRUE,
  id = new_step_id(rec, "lefse"),
  engine_args = list()
)
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires `add_model()` before `prep()`; the step can be configured before adding the model.
`kruskal.threshold`	numeric(1) The p-value for the Kruskal-Wallis Rank Sum Test (default 0.05).
`wilcox.threshold`	numeric(1) The p-value for the Wilcoxon Rank-Sum Test when 'blockCol' is present (default 0.05).
`lda.threshold`	numeric(1) The effect size threshold (default 2.0).
`subclassCol`	character(1) Optional column name in 'colData(expr)' indicating the blocks, usually a factor with two levels (e.g., 'c("adult", "senior")'; default NULL).
`assay`	The i-th assay matrix in the 'SummarizedExperiment' ('expr'; default 1).
`trim.names`	If 'TRUE' extracts the most specific taxonomic rank of organism.
`rarefy`	Boolean indicating if OTU counts must be rarefyed. This rarefaction uses the standard R sample function to resample from the abundance values in the otu_table component of the first argument, physeq. Often one of the major goals of this

procedure is to achieve parity in total number of counts between samples, as an alternative to other formal normalization procedures, which is why a single value for the sample.size is expected. If 'no_seed', rarefaction is performed without a set seed.

`id` Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names `taxa_id`, `contrast_id`, `comparison`, `contrast_type`, `var`, `effect`, `sum`, `.tax_order`, `.contrast_order` or `.effect_order`. Other identifiers, including names such as `.custom`, are retained unchanged.

`engine_args` Named lists of advanced arguments for the native `transform` or `fit` stage. Arguments managed by dar or exposed above cannot be overridden.

Value

An object of class `Recipe`

See Also

Other Diff taxa steps: `step_aldex2()`, `step_aldex3()`, `step_ancom()`, `step_corncob()`, `step_deseq()`, `step_linda()`, `step_maaslin()`, `step_rademu()`, `step_wilcox()`

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum")

rec_default <- step_lefse(rec)

# Require a larger LDA score.
rec_higher_lda <- step_lefse(rec, lda.threshold = 3)
```

---

step_linda *LinDA differential-abundance analysis*

Description

`step_linda()` fits bias-corrected linear or linear mixed-effects models with `MicrobiomeStat::linda()`. It requires a centralized model defined by `add_model()` and executes every contrast in that model's contrast plan.

Usage

```
step_linda(
  rec,
  prev_filter = 0,
  mean_abund_filter = 0,
  max_abund_filter = 0,
```

```
  winsorize = TRUE,
  outlier_pct = 0.03,
  adaptive = TRUE,
  zero_handling = c("pseudo-count", "imputation"),
  pseudo_count = 0.5,
  corr_cut = 0.1,
  p_adj_method = "BH",
  alpha = 0.05,
  n_cpus = 1L,
  rarefy = FALSE,
  id = new_step_id(rec, "linda"),
  engine_args = list()
)
```

Arguments

rec	An unprepared Recipe object. Execution requires add_model() before prep(); the step can be configured first.
prev_filter	Minimum feature prevalence retained for analysis.
mean_abund_filter	Minimum mean relative abundance retained.
max_abund_filter	Minimum maximum relative abundance retained.
winsorize	Whether high-abundance outliers are winsorized.
outlier_pct	Expected fraction of outliers used for winsorization.
adaptive	Whether LinDA chooses zero handling from depth-covariate correlations.
zero_handling	Zero treatment used when adaptive = FALSE: either "pseudo-count" or "imputation".
pseudo_count	Positive pseudo-count used by pseudo-count zero handling.
corr_cut	Significance cutoff used by adaptive zero handling.
p_adj_method	Multiple-testing correction accepted by p.adjust().
alpha	Adjusted p-value threshold used to classify significance.
n_cpus	Number of cores used by LinDA mixed-effects models.
rarefy	Whether counts are rarefied for this step.
id	Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names taxa_id, contrast_id, comparison, contrast_type, var, effect, sum, .tax_order, .contrast_order or .effect_order. Other identifiers, including names such as .custom, are retained unchanged.
engine_args	Named lists of advanced arguments for the native fit stage. Arguments managed by dar or exposed above cannot be overridden.

Details

With MicrobiomeStat 1.4, dar uses a local copy of the native function with the adaptive zero-handling capitalization error corrected. This restores the native choice of imputation when sequencing depth is associated with the design. The installed package, model fitting, winsorization and multiple-testing correction are unchanged.

Value

A Recipe object.

See Also

Other Diff taxa steps: step_aldex2(), step_aldex3(), step_ancom(), step_corncob(), step_deseq(), step_lefse(), step_maaslin(), step_rademu(), step_wilcox()

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Species")

rec_default <- step_linda(rec)

# Retain features present in at least 10% of samples.
rec_filtered <- step_linda(rec, prev_filter = 0.1)
```

---

step_maaslin *MaAsLin3 analysis*

Description

MaAsLin 3 finds associations between microbiome meta-omics features and complex metadata. It uses a unified framework to test for both abundance (using linear models) and prevalence (using logistic regression).

Usage

```
step_maaslin(
  rec,
  min_abundance = 0,
  min_prevalence = 0.1,
  min_variance = 0,
  normalization = "TSS",
  transform = "LOG",
  max_significance = 0.1,
  random_effects = NULL,
  correction = "BH",
  standardize = TRUE,
  reference = NULL,
  median_comparison_abundance = TRUE,
  rarefy = FALSE,
  id = new_step_id(rec, "maaslin"),
  engine_args = list()
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires add_model() before prep(); the step can be configured before adding the model.
min_abundance	The minimum abundance for each feature.
min_prevalence	The minimum percent of samples for which a feature is detected at minimum abundance.
min_variance	Keep features with variance greater than.
normalization	The normalization method to apply. Default: "TSS". Choices: "TSS", "CLR", "NONE".
transform	The transform to apply. Default: "LOG" (Base 2). Choices: "LOG", "PLOG", "NONE".
max_significance	The q-value threshold for significance.
random_effects	The random effects for the model (vector of character strings).
correction	The correction method for computing the q-value.
standardize	Apply z-score so continuous metadata are on the same scale.
reference	The factor to use as a reference for a variable with more than two levels provided as a string of 'variable,reference' semi-colon delimited.
median_comparison_abundance	Test abundance coefficients against a null value corresponding to the median coefficient for a metadata variable across the features. Recommended for relative abundance (default: TRUE).
rarefy	Boolean indicating if OTU counts must be rarefyed.
id	Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names taxa_id, contrast_id, comparison, contrast_type, var, effect, sum, .tax_order, .contrast_order or .effect_order. Other identifiers, including names such as .custom, are retained unchanged.
engine_args	Named lists of advanced arguments for the native fit or contrast stage. Arguments managed by dar or exposed above cannot be overridden.

Value

An object of class Recipe

See Also

Other Diff taxa steps: step_aldex2(), step_aldex3(), step_ancom(), step_corncob(), step_deseq(), step_lefse(), step_linda(), step_rademu(), step_wilcox()

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
```

```
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum")

rec_default <- step_maaslin(rec)

# Increase the minimum feature prevalence.
rec_filtered <- step_maaslin(rec, min_prevalence = 0.2)
```

`step_rademu` *radEmu differential-abundance analysis*

Description

Fit planned contrasts with `radEmu::emuFit()`. A centralized `add_model()` defines the fixed design and taxonomic analysis units. The default robust score test is recommended by radEmu; Wald testing is available explicitly.

Usage

```
step_rademu(
  rec,
  test = c("score", "wald"),
  cluster = NULL,
  max_significance = 0.05,
  p_adj_method = "BH",
  constraint_param = 0.1,
  seed = 130L,
  rarefy = FALSE,
  id = new_step_id(rec, "rademu"),
  engine_args = list()
)
```

Arguments

`rec`	An unprepared Recipe object.
`test`	Native test to report: `"score"` (default) or `"wald"`.
`cluster`	Optional single metadata column identifying dependent observations. `NULL` uses native independent-observation inference.
`max_significance`	Adjusted P value cutoff.
`p_adj_method`	Multiple-testing adjustment passed to `p.adjust()`.
`constraint_param`	Native pseudo-Huber smoothing parameter, default 0.1.
`seed`	Integer seed used for native fitting. Restores the caller's RNG.
`rarefy`	Whether to rarefy before fitting; default `FALSE`.
`id`	Unique step identifier.

`engine_args` Named list with an optional `fit` list of native arguments, for example `tolerance`, `maxit`, `maxit_null`, `alpha` and `remove_zero_comparison_pvals`. Solver controls can be passed directly in `fit`. Data, design, clustering, centring functions, fitted-model reuse, tested coordinates, test selection and result shape are managed by dar; `control` and `return_both_score_pvals` cannot be overridden.

Details

Effects are natural-log fold differences relative to the native pseudo-Huber centre across analysis taxa. They are not unqualified absolute-abundance changes or log2 fold changes. Each planned contrast is reparameterized and fitted with its own native centring constraint. This retains the native hypothesis rather than treating the nonlinear centre as a linear transformation of previously centred coefficients.

Supply `cluster` to identify a metadata column containing cluster labels. Labels are aligned with the common model cohort. Clustered inference uses radEmu's GEE framework, not a mixed model. Random-effect formulas such as `(1 | subject)` are incompatible and are never silently translated. For repeated observations, use a fixed formula and explicitly set `cluster = "subject"`. Missing cluster IDs are not dropped privately.

dar adjusts the selected native P values across returned taxa separately for each planned contrast. It does not adjust across all contrasts jointly. Missing native statistics remain missing, including native zero-comparison exclusions. Native convergence warnings are retained; diagnostics and effective settings are available through `get_execution()`.

The complete taxonomic analysis matrix is used in every fit. Counts are not split by taxon or restricted to the two groups named in a contrast. Score tests can be expensive because each taxon requires a restricted fit. Native numerical validation and errors are left to radEmu.

Value

A Recipe with the radEmu step added.

See Also

Other Diff taxa steps: `step_aldex2()`, `step_aldex3()`, `step_ancom()`, `step_corncob()`, `step_deseq()`, `step_lefse()`, `step_linda()`, `step_maaslin()`, `step_wilcox()`

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Genus") |>
  step_rademu()
# prep(rec) executes native score tests for every planned contrast and taxon.
```

`step_rarefaction` *Resample an OTU table such that all samples have the same library size.*

Description

Please note that the authors of phyloseq do not advocate using this as a normalization procedure, despite its recent popularity. Our justifications for using alternative approaches to address disparities in library sizes have been made available as an article in PLoS Computational Biology. See phyloseq_to_deseq2 for a recommended alternative to rarefying directly supported in the phyloseq package, as well as the supplemental materials for the PLoS-CB article and the phyloseq extensions repository on GitHub. Nevertheless, for comparison and demonstration, the rarefying procedure is implemented here in good faith and with options we hope are useful. This function uses the standard R sample function to resample from the abundance values in the otu_table component of the first argument, physeq. Often one of the major goals of this procedure is to achieve parity in total number of counts between samples, as an alternative to other formal normalization procedures, which is why a single value for the sample.size is expected. This kind of resampling can be performed with and without replacement, with replacement being the more computationally-efficient, default setting. See the replace parameter documentation for more details. We recommended that you explicitly select a random number generator seed before invoking this function, or, alternatively, that you explicitly provide a single positive integer argument as rngseed.

Usage

```
step_rarefaction(rec, id = new_step_id(rec, "rarefaction"))
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this Recipe.
`id`	A character string that is unique to this step to identify it.

Value

An object of class `Recipe`

Examples

```
data(metaHIV_phy)

recipe(metaHIV_phy) |>
  step_rarefaction()
```

`step_subset_taxa` *Subset taxa by taxonomic level*

Description

This is a convenience function around the subset_taxa function from the phyloseq package. It is designed to speed up subsetting complex experimental objects with one function call. In the case of run_subset_taxa, the subsetting will be based on the taxonomic level of each taxon. The taxa retained in the dataset are those where the taxonomic level matches the provided taxa.

Usage

```
step_subset_taxa(rec, tax_level, taxa, id = new_step_id(rec, "subset_taxa"))
```

Arguments

`rec`	A Recipe object. The step will be added to the sequence of operations for this Recipe.
`tax_level`	The taxonomic level for subsetting taxa.
`taxa`	The taxa to be retained in the dataset.
`id`	A character string that is unique to this step to identify it.

Details

The function subsets the taxa in the phyloseq object based on the provided taxonomic level and taxa. Only the taxa that match the provided taxa at the given taxonomic level are retained in the phyloseq object.

Value

A Recipe object that has been subsetted based on taxonomic level.

Note

This function modifies `rec` in place, you might want to make a copy of `rec` before modifying it if you need to preserve the original object.

See Also

`subset_taxa`

Examples

```
data(metaHIV_phy)

recipe(metaHIV_phy) |>
  step_subset_taxa(tax_level = "Kingdom", taxa = c("Bacteria", "Archaea")) |>
  prep()
```

step_wilcox *Wilcox analysis*

Description

Performs a wilcox test to determine features (be it Operational Taxanomic Unit (OTU), species, etc.) that are differentially abundant between two or more groups of multiple samples.

Usage

```
step_wilcox(
  rec,
  norm_method = "compositional",
  max_significance = 0.05,
  p_adj_method = "BH",
  rarefy = FALSE,
  id = new_step_id(rec, "wilcox"),
  engine_args = list()
)
```

Arguments

rec	A Recipe object. The step will be added to the sequence of operations for this Recipe. Execution requires add_model() before prep(); the step can be configured before adding the model.
norm_method	Transformation to apply. The options include: 'compositional' (ie relative abundance), 'Z', 'log10', 'log10p', 'hellinger', 'identity', 'clr', 'alr', or any method from the vegan::decostand function.
max_significance	The q-value threshold for significance.
p_adj_method	Character. Specifying the method to adjust p-values for multiple comparisons. Default is "BH" (Benjamini-Hochberg procedure).
rarefy	Boolean indicating if OTU counts must be rarefyed.
id	Unique, non-empty identifier for this step. DA step IDs must not use the reserved internal names taxa_id, contrast_id, comparison, contrast_type, var, effect, sum, .tax_order, .contrast_order or .effect_order. Other identifiers, including names such as .custom, are retained unchanged.
engine_args	Named lists of advanced arguments for the native test stage. Arguments managed by dar or exposed above cannot be overridden.

Value

An object of class Recipe

See Also

Other Diff taxa steps: step_aldex2(), step_aldex3(), step_ancom(), step_corncob(), step_deseq(), step_lefse(), step_linda(), step_maaslin(), step_rademu()

Examples

```
data(metaHIV_phy)

rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Phylum")

rec_default <- step_wilcox(rec)

# Use Holm's multiple-testing correction.
rec_holm <- step_wilcox(rec, p_adj_method = "holm")
```

---

steps_ids *Get step_ids from recipe*

Description

Get step_ids from recipe

Usage

```
steps_ids(rec, type = "all", include_skipped = FALSE)
```

Arguments

rec	A Recipe object.
type	character vector indicating the type class. Options c("all", "da", "prepro").
include_skipped	For a prepared modeled recipe, include DA steps that were declared but skipped as statistically incompatible.

Value

character vector

Examples

```
data(test_rec)

steps_ids(test_rec)

## Select differential-abundance or preprocessing steps
steps_ids(test_rec, type = "da")
steps_ids(test_rec, type = "prepro")
```

tax_table *Extract complete taxonomy from a recipe*

Description

Extract complete taxonomy from a recipe

Usage

```
tax_table(rec)
```

Arguments

rec	A Recipe or PrepRecipe object.

Value

A tibble with taxa_id followed by every taxonomic rank. Taxa are ordered according to phyloseq::taxa_names(). If the recipe has no taxonomy, an ID-only tibble is returned.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy)
tax_table(rec)
```

test_prep_rec *PrepRecipe for metaHIV_phy data*

Description

A prepared version of test_rec, fitted through the centralized model API at genus level. Results retain planned pairwise RiskGroup2 contrasts and method-specific effects. Original stored features remain available through otu_table(); tidy_abundance() returns their shared genus-level view. Unavailable native estimates remain missing; effect_plt() omits them rather than treating them as zero effects. Generated reproducibly by data-raw/metaHIV_recipes.R.

Usage

```
data("test_prep_rec")
```

Format

A PrepRecipe object.

test_rec *Recipe for metaHIV_phy data*

Description

An unprepared recipe for `metaHIV_phy`, with a centralized `~ RiskGroup2` model at genus level and `hts` as its reference. It includes microbial subsetting, feature-prevalence filtering, MaAsLin3 and DESeq2 steps.

Usage

```
data("test_rec")
```

Format

A Recipe object.

tidy_abundance *Extract abundances in the shared taxonomic analysis view*

Description

`tidy_abundance()` returns one row per taxon and sample. When a centralized model is defined, taxa are aggregated through its selected taxonomic rank, using the same identities as differential-abundance results. Without a centralized model, the stored features are returned without aggregation.

Usage

```
tidy_abundance(rec, taxa_ids = NULL, transform = "identity", metadata = NULL)
```

Arguments

`rec`	A Recipe or PrepRecipe object.
`taxa_ids`	Optional character vector of taxon identifiers to retain. IDs refer to the shared analysis view, not absorbed features. `NULL` retains every analysis taxon; `character(0)` returns no rows. Output keeps analysis taxon order rather than selector order.
`transform`	Either `"identity"` for stored abundances or `"compositional"` for within-sample fractions. Fractions use all shared analysis taxa as their denominator before `taxa_ids` selection. Zero-depth samples have missing fractions and produce a warning.
`metadata`	Character vector of sample metadata columns to include. `NULL` includes every column; `character(0)` includes none. Supply the original, unprefixed column names.

Details

This is a descriptive view of the data currently stored in the recipe. A prepared recipe includes common preprocessing and the shared model cohort; an unprepared recipe does not execute its configured steps. Neither view reproduces method-specific filtering, rarefaction or transformations, or adjusted model predictions.

Taxonomic columns are prefixed with `taxonomy_` and sample metadata columns with `metadata_`, so annotations cannot overwrite identifiers or abundances. Full lineage through the selected rank is retained; lower ranks removed by aggregation remain missing. Taxa sharing a label but not a lineage retain distinct `taxa_id` values.

Value

A tibble with `taxa_id`, `taxa`, `sample_id`, `abundance`, complete namespaced taxonomy and selected namespaced sample metadata. `taxa` uses the configured taxonomic label, or feature IDs when no single rank is configured. Sample metadata types are preserved. Rows follow analysis taxon order, then sample order.

Examples

```
data(metaHIV_phy)
rec <- recipe(metaHIV_phy) |>
  add_model(~ RiskGroup2, targets = "RiskGroup2", tax_level = "Genus")
abundances <- tidy_abundance(rec, metadata = "RiskGroup2")
head(abundances)

fractions <- tidy_abundance(
  rec, transform = "compositional", metadata = character(0)
)
```

`tidy_results` *Collect differential-abundance results in a tidy contract*

Description

`tidy_results()` combines the results of executed differential-abundance steps without changing the engine-specific tables stored in `rec@results`. Effect sizes are labelled with their engine-specific metric and must not be compared directly across incompatible metrics.

Usage

```
tidy_results(rec, steps = steps_ids(rec, "da"), significant_only = FALSE)
```

Arguments

`rec`	A PrepRecipe object.
`steps`	Character vector of executed differential-abundance step IDs to collect. By default, all executed DA steps are included.
`significant_only`	If `TRUE`, retain only rows classified as significant by their originating step.

Value

A deterministic tibble with one row per taxon, contrast and configured step. It contains taxon identifiers (`taxa_id`, `taxa`), contrast metadata (`contrast_id`, `comparison`, `contrast_type`, `var`, `numerator`, `denominator`, `at_var`, `at_level`), statistics (`effect_size`, `effect_metric`, `p_value`, `adj_p_value`, `significant`), the stable engine in `method`, and the configured instance in `step_id`.

Examples

```
data(test_prep_rec)
results <- tidy_results(test_prep_rec)
results

tidy_results(test_prep_rec, significant_only = TRUE)
```

Index

∗ **Bake steps**
  bake, 7
∗ **Diff taxa steps**
  step_aldex2, 28
  step_aldex3, 29
  step_ancom, 31
  step_corncob, 34
  step_deseq, 36
  step_lefse, 45
  step_linda, 46
  step_maaslin, 48
  step_rademu, 50
  step_wilcox, 54
∗ **datasets**
  metaHIV_phy, 18
  test_prep_rec, 56
  test_rec, 57
∗ **filter phy steps**
  step_filter_by_abundance, 39
  step_filter_by_prevalence, 40
  step_filter_by_rarity, 41
  step_filter_by_variance, 42
  step_filter_taxa, 43
∗ **rarefaction phy steps**
  step_rarefaction, 52
∗ **subset phy steps**
  step_subset_taxa, 53

abundance_plt, 4
add_model, 5
add_model(), *22, 23*, *28, 29*, *31, 32*, *35*, *37*, *45–47*, *49, 50*, *54*

bake, 7

contains_rarefaction, 8
cool, 8
corr_heatmap, 9

effect_plt, 10
effect_plt(), *56*
exclusion_plt, 11
export_steps, 12

filter_taxa, *40–43*
find_intersections, 13

get_execution, 14
get_execution(), *31*, *51*
get_model, 14
get_phy, 15

import_steps, 16
intersection_df, 17
intersection_plt, 17

list_or_NULL-class *(*phyloseq_or_null-class*)*, 22

metaHIV_phy, 18
MicrobiomeStat::linda(), *46*
mutual_plt, 19

otu_table, 20
otu_table(), *56*
overlap_df, 21

p.adjust(), *47*, *50*
phyloseq::filter_taxa(), *44*
phyloseq::sample_names(), *20*, *27*
phyloseq::taxa_names(), *20*, *56*
phyloseq_or_null-class, 22
prep, 22
prep(), *6*, *14*, *16*, *28*, *32*, *35*, *37*, *43–45*, *47*, *49*, *54*
PrepRecipe, *10*, *14*, *26*, *57, 58*
PrepRecipe(), *12*, *15, 16*
PrepRecipe-class, 23

radEmu::emuFit(), *50*
rand_id, 24

rarefaction_help, 24
Recipe, *26*, *47, 48*, *50*, *57*
Recipe *(*recipe*)*, 25
recipe, 25
Recipe(), *6*, *12*, *15, 16*
Recipe-class *(*phyloseq_or_null-class*)*, 22
recipe_qc, 26

sample_data, 27
show,PrepRecipe-method *(*phyloseq_or_null-class*)*, 22
step_aldex2, 28
step_aldex2(), *31*, *34*, *36*, *38*, *46*, *48, 49*, *51*, *55*
step_aldex3, 29
step_aldex3(), *29*, *34*, *36*, *38*, *46*, *48, 49*, *51*, *55*
step_ancom, 31
step_ancom(), *29*, *31*, *36*, *38*, *46*, *48, 49*, *51*, *55*
step_corncob, 34
step_corncob(), *29*, *31*, *34*, *38*, *46*, *48, 49*, *51*, *55*
step_deseq, 36
step_deseq(), *29*, *31*, *34*, *36*, *46*, *48, 49*, *51*, *55*
step_filter_by_abundance, 39
step_filter_by_abundance(), *41–44*
step_filter_by_prevalence, 40
step_filter_by_prevalence(), *40*, *42–44*
step_filter_by_rarity, 41
step_filter_by_rarity(), *40, 41*, *43, 44*
step_filter_by_variance, 42
step_filter_by_variance(), *40–42*, *44*
step_filter_taxa, 43
step_filter_taxa(), *40–43*
step_lefse, 45
step_lefse(), *29*, *31*, *34*, *36*, *38*, *48, 49*, *51*, *55*
step_linda, 46
step_linda(), *29*, *31*, *34*, *36*, *38*, *46*, *49*, *51*, *55*
step_maaslin, 48
step_maaslin(), *29*, *31*, *34*, *36*, *38*, *46*, *48*, *51*, *55*
step_rademu, 50
step_rademu(), *29*, *31*, *34*, *36*, *38*, *46*, *48, 49*, *55*
step_rarefaction, 52
step_subset_taxa, 53
step_wilcox, 54
step_wilcox(), *29*, *31*, *34*, *36*, *38*, *46*, *48, 49*, *51*
steps_ids, 55
subset_taxa, *53*

tax_table, 56
test_prep_rec, 56
test_rec, *56*, 57
tidy_abundance, 57
tidy_abundance(), *56*
tidy_results, 58