Package 'ALDEx3'

August 26, 2026

Title Linear Models for Sequence Count Data

Version 1.3.1

Description Provides scalable generalized linear and mixed effects models tailored for sequence count data analysis (e.g., analysis of 16S or RNA-seq data). Uses Dirichlet-multinomial sampling to quantify uncertainty in relative abundance or relative expression conditioned on observed count data.
Implements scale models as a generalization of normalizations which account for uncertainty in scale (e.g., total abundances) as described in Nixon et al. (2025) <doi:10.1186/s13059-025-03609-3> and McGovern et al. (2025) <doi:10.1101/2025.08.05.668734>.

License MIT + file LICENSE

Encoding UTF-8

Suggests rBeta2009, testthat (>= 3.0.0), lmtest, sandwich, knitr, rmarkdown

Config/testthat/edition 3

Imports purrr, lme4, lmerTest, reformulas, parallel, MASS, nlme, abind, matrixStats, methods, stats, TMB

LinkingTo TMB, RcppEigen, Matrix

Depends R (>= 3.5)

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

SystemRequirements C++17

NeedsCompilation yes

Author Justin Silverman [aut, cre], Greg Gloor [aut], Kyle McGovern [aut, ctb]

Maintainer Justin Silverman <JustinSilverman@psu.edu>

Repository CRAN

Date/Publication 2026-08-26 15:40:02 UTC

Contents

aldex . . . 2
aldex.effect . . . 5
aldex.lm.sim.clr . . . 6
aldex.mem.sim . . . 7
aldex.plot . . . 8
aldex.pvals . . . 9
blmm . . . 10
clr.sm . . . 11
coefficient.sm . . . 12
cohensd . . . 13
combine.streams . . . 14
fflm . . . 15
gut_crohns_data . . . 15
miniclo . . . 16
oral_mouthwash_data . . . 17
req . . . 17
rLogDirichlet . . . 18
sample.sm . . . 18
sr.mem . . . 20
summary.aldex . . . 21
tss.sm . . . 21

Index **23**

`aldex` *ALDEx3 Linear Models*

Description

ALDEx3 Linear Models

Usage

```
aldex(
  Y,
  X,
  data = NULL,
  method = "lm",
  nsample = 2000,
  scale = NULL,
  streamsize = 8000,
  n.cores = detectCores() - 1,
  return.pars = c("X", "estimate", "std.error", "p.val", "p.val.adj", "logComp",
    "logScale"),
  p.adjust.method = "BH",
  test = "t.HC3",
```

```
  onesided = FALSE,
  ...
)
```

Arguments

Y	an (D x N) matrix of sequence count, N is number of samples, D is number of taxa or genes
X	either a formula (in which case DATA must be non-null) or a model matrix of dimension P x N (P is number of linear model covariates). If a formula is passed it should not include the target Y, e.g., should simply be "~condition-1" (note the lack of the left hand side). If using lme4, this should be the formula including random effects.
data	a data frame for use with formula, must have N rows
method	(default lm) The regression method; "lm": linear regression, "lme4": linear mixed effects regression with lme4; "nlme": linear mixed effects models with nlme, REQUIRES "random" argument representing random effects be passed into aldex function, can also pass "correlation" as argument (see nlme documentation for how to use "correlation" argument); or "blmm": an ALDEx3-specific approximate mixed-effects engine that uses one batched profiled mixed-model anchor fit per feature, draw-specific local covariance updates, and exact conditional fixed-effect solves. The approximation is only in the variance-component optimisation step. See the mixed-effects vignette for details, validation guidance, and the runtime comparison with exact lme4.
nsample	number of monte carlo replicates
scale	A scale-model function, such as clr.sm, tss.sm, sample.sm, or coefficient.sm; alternatively, an N x nsample matrix of log2-scale draws. Additional arguments for a scale-model function are passed through .... See "Writing a custom scale model" in the Examples section for the required interface.
streamsize	(default 8000) memory footprint (approximate) at which to use streaming. This should be thought of as the number of Mb for each streaming chunk. If D*N*nsample*8/1000000 is less than streamsize then no streaming will be performed. Note, to conserve memory, samples from the Dirichlet and scale models will not be returned if streaming is used. Streaming can be turned off by setting streamsize=Inf.
n.cores	(default detectCores()-1) If method is 'lme4' or 'blmm', use this many cores for feature-level parallelism across taxa/features. For method = "blmm", parallelism is across features, not across Monte Carlo draws within a feature.
return.pars	what results should be returned, see return section below.
p.adjust.method	(default BH) The method for multiple hypothesis test correction. See p.adjust for all available methods.
test	(default t.HC3), "t", t test is performed for each covariate (fast); "t.HC0" Heteroskedasticsity-Robust Standard Errors used (HC0; White's; slower); "t.HC3" (default) Heteroskedasticsity-Robust Standard Errors used (HC3; unlike HC0, this includes a leverage adjustment and is better for small sample sizes or when there are data with high leverage; slowest). To learn more about these, loko at Long and Ervin (2000) Using

	Heteroscedasticity Consistent Standard Errors in the Linear Regression Model, The American Statistician.
onesided	(default: FALSE) if sided return p-values for two-sided test. Otherwise if "lower" or "upper" return one-sided test corresponding to test that estimate is negative or positive respectively.
...	Additional scale-model arguments, such as gamma, s.mu, or c.mu. For method = "nlme", this may also include random or correlation.

Details

Sequence counts determine the relative composition of each sample, but not its total abundance or scale. The scale argument tells ALDEx3 what is known about that missing quantity and how uncertain that knowledge is.

Choose a scale model based on the information available:

- Use sample.sm for external measurements such as flow cytometry, qPCR, or spike-ins for individual samples.
- Use coefficient.sm when prior knowledge concerns a group difference, treatment effect, time trend, or another model term.
- Use clr.sm when CLR normalization is a reasonable center but its implied scale differences are uncertain.
- Use tss.sm when the starting assumption is no systematic scale difference between groups.
- Write a custom scale-model function when the built-in models do not match the information available for the study. The "Writing a custom scale model" example below describes the required function interface and N x nsample output.

Value

a list with elements controled by parameter resturn.pars. Options include: - X: P x N covariate matrix - estimate: (P x D x nsample) array of linear model estimates - std.error: (P x D x nsample) array of standard error for the estimates - p.val: (P x D) matrix, unadjusted p-value for two-sided t-test - p.val.adj: (P x D) matrix, p-value for two-sided t-test adjusted according to p.adj.method - logScale: (N x S) matrix, samples of the log scale from the scale model - logComp: (D x N x S) array, samples of the log composition from the multinomial-Dirichlet - streaming: boolean, detnote if streaming was used. - random.effects (Pr x N x S): if using mixed effects models, return all Pr random-effects covariance terms and the residual variance. For method = "blmm", correlated random-effect covariance terms are returned when present, matching the lme4 naming convention as closely as possible. Note, logScale and logComp are not returned if streaming is active.

Author(s)

Justin Silverman, Kyle McGovern

Examples

```
Y <- matrix(1:110, 10, 11)
condition <- c(rep(0, 5), rep(1, 6))
data <- data.frame(condition=condition)
```

```
## Use CLR as the center and allow coefficient-scale uncertainty of SD 0.5.
res <- aldex(Y, ~condition, data, nsample=2000, scale=clr.sm, gamma=0.5)

## Writing a custom scale model
## A custom model must return one log2-scale draw per sample and Monte
## Carlo replicate. Here, external_scale supplies a known value per sample.
known_scale <- function(X, logComp, external_scale) {
  nsample <- dim(logComp)[3]
  matrix(rep(external_scale, nsample), nrow=ncol(X), ncol=nsample)
}
res <- aldex(Y, ~condition, data, nsample=2000, scale=known_scale,
             external_scale=rep(0, ncol(Y)))
```

`aldex.effect` *ALDEx3 Effect Diagnostics*

Description

ALDEx2-inspired effect diagnostics for an ALDEx3 binary contrast

Usage

```
aldex.effect(object, contrast)
```

Arguments

`object`	An object returned by `aldex`.
`contrast`	The exact model coefficient or unambiguous binary data column to summarize.

Details

`aldex.effect` is a helper for effect-size diagnostics that Greg Gloor and ALDEx2 users commonly inspect when evaluating pairwise effects. It is not a replacement for `summary.aldex`, which reports model estimates, standard errors and adjusted p-values. This function summarizes a single binary ALDEx3 model contrast using Cohen's-d-based Monte Carlo diagnostics.

For a selected contrast row `k`, feature `d`, Monte Carlo draw `s`, and binary group indicators `x = object$X[k, ]`, let `B[d, s]` be the fitted ALDEx3 coefficient stored in `object$estimate[k, d, s]`. Let `logW[d, i, s] = object$logComp[d, i, s] + object$logScale[i, s]` be the reconstructed log abundance for sample `i`. `aldex.effect` requires `logComp` and `logScale`, so it is unavailable when `aldex` streams large jobs and omits those arrays. For each feature and Monte Carlo draw, the pooled within-group variance is `((n0 - 1) * var(logW[d, x == 0, s]) + (n1 - 1) * var(logW[d, x == 1, s])) / (n0 + n1 - 2)`, where `n0` and `n1` are the group sizes.

The returned columns are calculated as follows:

- `estimate`: the mean of `B[d, s]` over Monte Carlo draws.
- `pooled.SD`: the mean of the pooled within-group standard deviation, `sqrt(pooled variance)`, over Monte Carlo draws.

- cohens.d: the mean of B[d, s] / pooled.SD[d, s] over Monte Carlo draws. This is Cohen's-d-based and is not the same as the historical robust ALDEx2 effect statistic based on between/within dispersion.
- overlap: an ALDEx2-style directional uncertainty diagnostic. It counts the Monte Carlo Cohen's d draws below and above zero, smooths those two counts with a 0.5 pseudocount, converts them with the ALDEx/Aitchison mean, and returns the smaller smoothed sign probability. Values near 0 indicate that nearly all Monte Carlo effect draws have the same sign. Values near 0.5 indicate uncertainty about direction. overlap is not a p-value, a confidence interval probability, or a literal area of overlap between two density curves.

Value

A data.frame with columns parameter, entity, estimate, pooled.SD, cohens.d, and overlap.

Author(s)

Greg Gloor, Justin Silverman

`aldex.lm.sim.clr` *Simulation function soley for testing and exploring ALDEx3, Truth is in CLR Coordinates.*

Description

Not designed to create realistic data. Does not add any noise to linear regression! True W is in CLR Corrdinates

Usage

```
aldex.lm.sim.clr(D = 10, N = 11, P = 2, depth = 10000)
```

Arguments

D	number of taxa/genes
N	number of samples
P	number of covariates
depth	sum of counts for each multinomial draw

Value

a list with elements Y, X, W, and Lambda

Author(s)

Justin Silverman

`aldex.mem.sim` *Simulation for testing mixed effects models.*

Description

Includes random intercpt, option to include random slope, second random intercept, and time-correlation.

Usage

```
aldex.mem.sim(
  D,
  days,
  subjects,
  depth = 10000,
  location = 1,
  random_slope = FALSE,
  corr = 0,
  rho_ar1 = 0,
  sd_resid = 0.1
)
```

Arguments

`D`	number of taxa/genes
`days`	num days (i.e., repeated measurements) for each subject
`subjects`	num of subjects to simulate
`depth`	sum of counts for each multinomial draw
`location`	second random intercept, if 0 ignore, else simulate num of locations
`random_slope`	If true, simulate a random slope for each subject.
`corr`	The correlation between slope/random intercept
`rho_ar1`	The ar1 time-correlation, if 0, don't simulate
`sd_resid`	The residual error time-correlation.

Author(s)

Kyle McGovern

`aldex.plot` *Plot Method for ALDEx3 objects with pairwise comparisons*

Description

Simple plots for an ALDEx3 object

Usage

```
aldex.plot(
  object,
  plot = c("volcano", "effect", "MA", "water"),
  contrast = NULL,
  threshold = 0.05,
  min.diff = 0.5,
  cohen = 0.5,
  sig.col = rgb(1, 0, 0, 0.5),
  water.show = 5,
  water.col = c("red", "blue"),
  water.names = TRUE,
  ...
)
```

Arguments

`object`	An object of class `aldex`
`plot`	type of plot (default='volcano')
`contrast`	the name of the comparison to plot, must be provided
`threshold`	FDR significance threshold (default=0.05)
`min.diff`	(default=0.5) used for MA (display), and waterfall (cutoff) plots
`cohen`	Cohen's d threshold for effect plot (slope)
`sig.col`	color for significant features
`water.show`	number features to show in waterfall plot
`water.col`	vector of colors for waterfall plot
`water.names`	logical, show names of features for waterfall plot
`...`	additional plot parameters (cex)

Details

Provides volcano, effect, MA and waterfall plots from an ALDEx3 object.

This method plots combinations of adjusted p-values from `object$p.val.adj`, posterior estimates, standard errors and log abundance values averaged across Monte Carlo samples. The result is returned as a single plot of the desired type. Effect plots use the ALDEx2-inspired diagnostics returned by `aldex.effect`. The contrast must identify exactly one binary model coefficient. Effect diagnostics and MA plots require `logComp` and `logScale`, which are not returned when `aldex` streams large jobs.

Value

the desired plot

Author(s)

Greg Gloor

`aldex.pvals` *Calculate p-values adjusting for changes in sign as described by Nixon et al. (2024) in Beyond Normalization: Incorporating Scale Uncertainty in Microbiome and Gene Expression Analysis (internal only)*

Description

Calculate p-values adjusting for changes in sign as described by Nixon et al. (2024) in Beyond Normalization: Incorporating Scale Uncertainty in Microbiome and Gene Expression Analysis (internal only)

Usage

```
aldex.pvals(p.lower, p.upper, p.adjust.method, onesided = FALSE)
```

Arguments

`p.lower`	A P x D x S matrix for P covariates, D taxa/genes, and S monte carlo samples representing the lower tail p.values
`p.upper`	A P x D x S matrix for P covariates, D taxa/genes, and S monte carlo samples representing the upper tail p.values
`p.adjust.method`	An adjutment method for p.adjust
`onesided`	(default: FALSE) if sided return p-values for two-sided test. Otherwise if "lower" or "upper" return one-sided test corresponding to test that estimate is negative or positive respectively.

Value

A list with P x D matrices with the non-adjusted and adjusted p-values.

Author(s)

Justin Silverman, Kyle McGovern

References

Nixon G, Gloor GB, Silverman JD (2025). "Incorporating scale uncertainty in microbiome and gene expression analysis as an extension of normalization". Genome Biology. doi:10.1186/s13059-025036093

`blmm` *Fast approximate mixed-effects engine for ALDEx3.*

Description

Fast approximate mixed-effects engine for ALDEx3.

Usage

```
blmm(logW, formula, data, n.cores = 1L)
```

Arguments

`logW`	numeric array (N x D x S)
`formula`	lme4 mixed-effects formula
`data`	data.frame for formula evaluation
`n.cores`	number of worker processes for feature-level parallelism. If `n.cores > 1`, features are distributed across workers; the per-feature draw loop remains serial.

Details

Internal engine called by `aldex` when `method = "blmm"`.

Uses an `lme4`-style profiled Gaussian LMM formulation. One batched anchor REML problem is fit per feature across all S MC draws; draw-specific local covariance updates are obtained via a single Newton step; exact conditional GLS fixed-effect solves are then run per draw. The approximation is only in the variance-component step — fixed effects and standard errors are exact conditional on the updated covariance parameters.

Features for which the approximate path fails are silently re-fit with the exact `lme4` engine and a consolidated warning is issued naming the affected features and their error messages.

Value

named list of arrays (P x D x S): estimate, std.error, df, p.lower, p.upper, and (Pr x D x S) random.eff — same layout as `sr.mem`.

`clr.sm` *CLR-based scale model*

Description

Use this model when centered log-ratio (CLR) normalization is a reasonable starting point, but you do not want to assume that it recovers the sample scales exactly. For example, in a case-control study, CLR may provide a plausible center while `gamma` allows the true difference in total abundance between cases and controls to depart from that center.

Usage

```
clr.sm(X, logComp, gamma = 0.5)
```

Arguments

`X`	Model matrix passed automatically by `aldex()`. Rows are model coefficients and columns are samples.
`logComp`	Monte Carlo log-compositions passed automatically by `aldex()`, with dimensions `features` x `samples` x `nsample`.
`gamma`	Non-negative scalar. Standard deviation of the Gaussian random shift that relaxes the CLR assumption about scale. `gamma = 0` yields the fixed CLR assumption. Larger values express less confidence in the CLR-implied differences between sample scales.

Details

Set `gamma = 0` to use the CLR-implied scales without additional uncertainty. A value such as `gamma = 0.5` allows each scale-model coefficient to vary with SD 0.5 on the log2 scale. Larger values express less confidence in the CLR assumption.

The uncertainty applies to model coefficients, not independently to every sample. Samples with the same covariate values therefore receive the same random shift in a Monte Carlo draw. This differs from `sample.sm`, which can draw a separate scale for each sample.

Value

A numeric matrix of dimension `N` x `nsample` giving Monte Carlo samples of the log-scale for each sample (rows) and each Monte Carlo draw (columns).

Author(s)

Justin Silverman

References

Nixon G, Gloor GB, Silverman JD (2025). "Incorporating scale uncertainty in microbiome and gene expression analysis as an extension of normalization". Genome Biology. doi:10.1186/s13059-025036093

Examples

```
# Allow moderate uncertainty around CLR-implied scale differences.
Y <- matrix(seq_len(60), nrow = 10)
condition <- factor(c("control", "control", "control",
                      "case", "case", "case"))
fit <- aldex(Y, ~ condition, data.frame(condition), nsample = 20,
             scale = clr.sm, gamma = 0.5)
```

`coefficient.sm` *Scale model using prior information about groups or treatments*

Description

Use this model when prior information describes a treatment, group, time, or other model coefficient rather than an individual sample. Samples with the same covariate values share the same scale shift in each Monte Carlo draw.

Usage

```
coefficient.sm(
  X,
  logComp,
  c.mu = NULL,
  c.cor = NULL,
  c.sd = NULL,
  c.cov = NULL
)
```

Arguments

`X`	Model matrix passed automatically by `aldex()`. Rows are model coefficients and columns are samples.
`logComp`	Monte Carlo log-compositions passed automatically by `aldex()`. This function uses the third dimension to determine the number of draws.
`c.mu`	Expected log2 scale change for each model coefficient, in the same order as the rows of `X`. Must have length `P`.
`c.cor`	Deprecated `P` x `P` covariance matrix for the fixed-effect coefficients. Use `c.cov` instead. Retained until ALDEx3 2.0.0.
`c.sd`	SD of the log2 scale change for each model coefficient. Must have length `P`; coefficients are treated as independent.
`c.cov`	A `P` x `P` covariance matrix for the model coefficients' log2 scale changes. Use this instead of `c.sd` when coefficient uncertainties are correlated.

Details

Suppose an effect plot from a control-versus-treatment experiment is centered near -8, even though prior knowledge suggests that most features should not change. One way to represent that knowledge is to shift the treatment samples upward by 8 log2 units relative to the controls. With control as the reference level, use `c.mu = c(0, 8)`: the first value leaves the common intercept unchanged, and the second adds 8 to treatment. Setting `c.sd = c(0, 0.5)` expresses an SD of 0.5 around that treatment shift. If the plot were centered near +8, use -8 instead. Adding 8 to both groups would not recenter the plot because it would leave their difference unchanged.

More generally, `c.mu` gives the expected log2 scale change for each model coefficient and `c.sd` gives its SD. Use `c.cov` when the coefficient uncertainties are correlated. For each Monte Carlo draw, ALDEx3 samples a coefficient vector and applies it to the model matrix `X`; mathematically, $b^{(m)} \sim N(c.mu, Sigma)$ and the sample scales are $b^{(m)T} X$.

Exactly one of `c.sd` or `c.cov` must be provided. The deprecated `c.cor` argument may be used in place of `c.cov` during the compatibility period.

Value

A numeric matrix of dimension `N x nsample` giving Monte Carlo draws of the log2 scale for each sample (rows) across `nsample` draws (columns).

Author(s)

Kyle McGovern

Examples

```
# Three control samples followed by three treatment samples.
condition <- factor(
  c("control", "control", "control", "treatment", "treatment", "treatment"),
  levels = c("control", "treatment")
)
X <- t(model.matrix(~ condition))
logComp <- array(0, c(2, 6, 10))
logScale <- coefficient.sm(
  # Shift treatment upward by 8, with SD 0.5 around that shift.
  X, logComp, c.mu = c(0, 8), c.sd = c(0, 0.5)
)
```

---

cohensd *Cohen's D*

Description

Function to compute cohensd on the results provided by the aldex function

Usage

```
cohensd(m, var)
```

Arguments

m	the output of a call to aldex
var	if aldex was called with X being a pre-computed model matrix, this var should be an integer corresponding to a binary covariate indicating the desired effect size to calculate (an effect size between two groups indicated by the binary covariate). For example, if the third covariate in the model is an indicator denoting health (0) and disease (1) then set var=3. In contrast, if X was a formula (in which case the data argument should have been specified) then var can be set to the unquoted or quoted name of the binary condition variable (e.g., var=condition or var="condition").

Details

WARNING: this function is experimental and requires users read the documentation fully.

Value

A (D x nsample)-matrix of Cohen's d statistics for the variable of interest.

Author(s)

Justin Silverman

---

combine.streams *Combine output from aldex streams (internal only)*

Description

Takes a list l, where every element is itself a list of 3D arrays. Each array should have matching first and second dimentions. This function combines those arrays along the third dimension.

Usage

```
combine.streams(l)
```

Arguments

l	a list, where every element is itself a list of 3D arrays.

Value

a list of 3D arrays

Author(s)

Justin Silverman

fflm *Freaking Fask Linear Models*

Description

Tailored for ALDEx3 where covariates are shared between massive numbers of linear regressions where only Y is changing.

Usage

```
fflm(Y, X, test = "t.HC3")
```

Arguments

Y	a numeric array (N x D X S) where D is the number of taxa/genes, N is the number of samples, and S is the number of posterior samples
X	a numeric matrix (N x P) where P is number of covariates
test	(default t.HC3), "t", t test is performed for each covariate (fast); "t.HC0" Heteroskedasticsity-Robust Standard Errors used (HC0; White's; slower); "t.HC3" (default) Heteroskedasticsity-Robust Standard Errors used (HC3; unlike HC0, this includes a leverage adjustment and is better for small sample sizes or when there are data with high leverage; slowest). To learn more about these, loko at Long and Ervin (2000) Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model, The American Statistician.

Value

A list of (P x D x S)-arrays with the OLS point estimates, the standard errors, and the two-sided p-values for each coefficient (P), of each model fit to each taxa (D) and each posterior sample (S)

Author(s)

Justin Silverman

gut_crohns_data *Gut Crohn's microbiome dataset (list)*

Description

This dataset contains a matrix and a data frame: genus-level microbiome profiles and corresponding sample metadata from a Crohn's disease case–control cohort. The dataset is used in examples and vignettes throughout the package.

Usage

```
gut_crohns_data
```

Format

A named list of one matrix and a dataframe:

`counts` A matrix with read counts with 195 rows and 95 columns)

`metadata` A data frame with 95 rows and 7 columns containing subject-level covariates.

Details

The `counts` matrix has one row per sample and one column per genus. The `metadata` data frame has one row per sample with metadata, critically Health.status either CD or Control, and Average cell count per gram frozen feces.

Source

Vandeputte D, Falony G, Vieira-Silva S, Wang J, Sailer M, Theis S, Raes J (2017). "Quantitative microbiome profiling links gut community variation to microbial load". Nature, 551, 507–511. doi:10.1038/nature24460

`miniclo` *Closure operation*

Description

Closure operation

Usage

```
miniclo(X)
```

Arguments

`X`	should be a D x N matrix, closes along the D dimension

Value

D x N matrix with columns summing to 1

Author(s)

Justin Silverman

oral_mouthwash_data *Oral microbiome perturbation dataset (list)*

Description

This dataset contains a matrix and a data frame: genus-level microbiome profiles and corresponding sample metadata from an oral microbiome perturbation study. 28 participants' oral microbiomes were measured before, 15 minutes after, and 2 hours after perturbation with either a water control, antiseptic mouthwash, alchohol-free mouthwash, or soda. The dataset is used in examples and vignettes throughout the package.

Usage

```
oral_mouthwash_data
```

Format

A named list of one matrix and a dataframe:

`counts` A matrix with read counts with 116 rows and 81 columns)

`metadata` A data frame with 81 rows and 38 columns containing sample-level covariates.

Details

The `counts` matrix has one row per sample and one column per genus. The `metadata` data frame has one row per sample with metadata, critically the participant ID, flow cytometry average (and replicate) cells, time_c (time points), and treat (the perturbations)

Source

Marotz C, Morton JT, Navarro P, Coker J, Belda-Ferre P, Knight R (2021). "Quantifying live microbial load in human saliva samples over time reveals stable composition and dynamic load". mSystems, 6(3), e01182-21. doi:10.1128/mSystems.0118220

req *Test object contains elements*

Description

Test object contains elements

Usage

```
req(obj, names)
```

Arguments

obj	object (list type)
names	names of elements required to be in the list

Author(s)

Justin Silverman

---

rLogDirichlet *Function for sampling Dirichlet random variables (base-2 normalized via log-sum-exp for stability)*

Description

Function for sampling Dirichlet random variables (base-2 normalized via log-sum-exp for stability)

Usage

```
rLogDirichlet(n, alpha)
```

Arguments

n	number of samples
alpha	D-vector of Dirichlet parameters

Value

a D x n matrix of samples

Author(s)

Justin Silverman

---

sample.sm *Per-sample scale model using external information*

Description

Use this model when each sample has its own external scale measurement, such as a flow-cytometry cell count, qPCR concentration, or spike-in estimate. Put the log2 measurement for each sample in `s.mu`. Use `s.sd` to describe the measurement uncertainty for each sample.

Usage

```
sample.sm(
  X,
  logComp,
  s.mu = NULL,
  s.var = NULL,
  s.cor = NULL,
  s.sd = NULL,
  s.cov = NULL
)
```

Arguments

X	Model matrix passed automatically by aldex(). This function uses its number of columns to determine the number of samples.
logComp	Monte Carlo log-compositions passed automatically by aldex(). This function uses the third dimension to determine the number of draws.
s.mu	Expected log2 scale for each sample, in the same order as the columns of the count matrix. Must have length N.
s.var	Deprecated numeric vector of per-sample log2-scale variances. Use s.sd = sqrt(s.var) instead. Retained until ALDEx3 2.0.0 for backward compatibility.
s.cor	Deprecated N x N covariance matrix. Despite its name, this argument has always represented covariance rather than correlation. Use s.cov instead. Retained until ALDEx3 2.0.0.
s.sd	SD of the log2 scale for each sample. Must have length N. Use this when measurement errors are independent across samples.
s.cov	An N x N covariance matrix for the samples' log2 scales. Use this instead of s.sd when measurement errors are correlated.

Details

For example, if four samples have estimated total cell counts of 1e8, 2e8, 8e7, and 1.5e8, set s.mu = log2(c(1e8, 2e8, 8e7, 1.5e8)). If each measurement has an SD of 0.5 on the log2 scale, set s.sd = rep(0.5, 4).

Use s.sd when measurement errors are independent. Use s.cov instead when errors are correlated, for example because several samples share a calibration standard. The diagonal of s.cov contains variances, and its off-diagonal entries contain covariances. Thus independent SDs of 0.5 can also be written as s.cov = diag(0.5^2, 4).

Supply exactly one of s.sd or s.cov. With s.sd, each sample receives its own independent scale draw. Samples in the same group do not automatically share uncertainty as they do in coefficient-based models.

Value

A numeric matrix of dimension N x nsample giving Monte Carlo draws of the log2 scale for each sample (rows) across nsample draws (columns).

Author(s)

Kyle McGovern

Examples

```
# External total-cell estimates for four samples.
cell_count <- c(1e8, 2e8, 8e7, 1.5e8)
X <- matrix(1, 1, 4)
logComp <- array(0, c(2, 4, 10))
logScale <- sample.sm(
  X, logComp, s.mu = log2(cell_count), s.sd = rep(0.5, 4)
)
```

---

sr.mem *Implementation of SR-MEM: scale-reliant mixed effects models.*

Description

Implementation of SR-MEM: scale-reliant mixed effects models.

Usage

```
sr.mem(logW, formula, data, n.cores, method, mem.args)
```

Arguments

logW	a numeric array (N x D X S) where D is the number of taxa, N is the number of samples, and S is the number of posterior samples
formula	an lme4 formula with fixed and random effects for lmer
data	A data.frame with the random and fixed effects items in formula
n.cores	The number of cores for parallelization. If n.cores=1, no parallelization is used
method	The mem method to use: either nlme or lme4
mem.args	Additional arguments including random or correlation

Value

A list of (P x D x S)-arrays with the fixed effect point estimates, the standard errors, degrees of freedom and the lower and upper p-values for each coefficient (P), of each model fit to each taxa (D) and each posterior sample (S) and a (Pr x D x S)-array with (Pr) random effects.

Author(s)

Kyle McGovern

summary.aldex *Summary Method for ALDEx3 Objects*

Description

Summarize an ALDEx3 result object

Usage

```
## S3 method for class 'aldex'
summary(object, ignore.intercept = TRUE, ...)
```

Arguments

object	An object of class aldex
ignore.intercept	(default=TRUE), ignore intercept when creating summary table
...	Additional arguments (currently ignored).

Details

Provides a summary of the adjusted p-values, estimates, and standard errors from an ALDEx3 result object.

This method extracts adjusted p-values from object$p.val.adj, along with posterior estimates and standard errors averaged across Monte Carlo samples. The result is returned as a long-format data.frame suitable for downstream analysis or visualization.

Value

A data.frame with columns parameter, entity, p.val.adjusted, estimate, and std.error.

Author(s)

Justin Silverman

tss.sm *TSS-centered scale model*

Description

Total-sum scaling (TSS) starts from the assumption that total scale does not change systematically between experimental groups. Use this model when that is a reasonable starting point but you want to express uncertainty about it. For example, if a treatment is expected to redistribute features without changing total biomass, TSS centers the treatment-versus-control scale difference at zero.

Usage

```
tss.sm(X, logComp, gamma = 0.5)
```

Arguments

X	Model matrix passed automatically by aldex(). Rows are model coefficients and columns are samples.
logComp	Monte Carlo log-compositions passed automatically by aldex(). This function uses the third dimension to determine the number of draws.
gamma	Non-negative scalar. Standard deviation of the Gaussian random shift applied to the scale-model coefficients (in log2 space). gamma = 0 implies no scale uncertainty (all draws are centered at zero effect); larger values allow greater departures from the TSS-centered assumption.

Details

Set gamma = 0 to enforce no scale difference. Setting gamma = 0.5 keeps the expected difference at zero but allows each scale-model coefficient to vary with SD 0.5 on the log2 scale. Larger values express less confidence that total scale is unchanged.

ALDEx3 applies this uncertainty to model coefficients: for each draw, $b^{(m)} \sim N(0, \gamma^2 I)$, and the sample scales are $b^{(m)T} X$. Samples with the same covariate values therefore share a random shift. This differs from sample.sm, which can draw independent uncertainty for every sample.

Value

A numeric matrix of dimension N x nsample giving Monte Carlo draws of the log2 scale for each sample (rows) across nsample draws (columns).

Author(s)

Justin Silverman

References

Nixon G, Gloor GB, Silverman JD (2025). "Incorporating scale uncertainty in microbiome and gene expression analysis as an extension of normalization". Genome Biology. doi:10.1186/s13059-025036093

Examples

```
# Center the group-scale difference at zero, with moderate uncertainty.
Y <- matrix(seq_len(60), nrow = 10)
condition <- factor(c("control", "control", "control",
                      "treatment", "treatment", "treatment"))
fit <- aldex(Y, ~ condition, data.frame(condition), nsample = 20,
             scale = tss.sm, gamma = 0.5)
```

Index

∗ **datasets**
gut_crohns_data, 15
oral_mouthwash_data, 17

aldex, 2, *10*
aldex.effect, 5
aldex.lm.sim.clr, 6
aldex.mem.sim, 7
aldex.plot, 8
aldex.pvals, 9

blmm, 10

clr.sm, *4*, 11
coefficient.sm, *4*, 12
cohensd, 13
combine.streams, 14

fflm, 15

gut_crohns_data, 15

miniclo, 16

oral_mouthwash_data, 17

req, 17
rLogDirichlet, 18

sample.sm, *4*, *11*, 18, *22*
sr.mem, 20
summary.aldex, 21

tss.sm, *4*, 21