Package: power.nb (via r-universe)

June 16, 2026

Title Power and Sample Size Calculation for Differential Abundance Microbiome Studies

Version 0.1.0

Description Provides functions for estimating statistical power and required sample sizes in differential abundance microbiome studies using negative binomial models. The methods are based on Agronah and Bolker (2025) <doi:10.1371/journal.pone.0318820>. The package includes tools for simulation-based power analysis and sample size estimation using generalized additive models (GAMs), and visualization utilities for exploring the relationship between power, effect size, abundance, and sample size.

License MIT + file LICENSE

URL https://michaelagronah.com/power.nb/

BugReports https://github.com/magronah/power.nb/issues

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports DESeq2, fitdistrplus, rlang, minpack.lm,foreach,tibble, purrr, mixtools, scam, ggplot2, ggrastr, metR, DEoptim, latex2exp, parallel,stats,doParallel

Suggests rmarkdown, knitr, kableExtra, tidyverse, patchwork, dplyr, rlist

RoxygenNote 7.3.3

Config/pak/sysreqs libabsl-dev libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

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

Date/Publication 2026-06-12 00:48:53 UTC

RemoteUrl https://github.com/magronah/power.nb

RemoteRef HEAD

RemoteSha 26d33e296987930e8752c1cfda18a20bf17358df

Contents

contour_plot_fun . . . 2
countdata_sim_fun . . . 3
deseq_fun_est . . . 5
deseqfun . . . 6
dispersion_fit . . . 8
dispersion_fun . . . 9
dnormmix . . . 10
dnormmix0 . . . 11
filter_low_count . . . 11
gam_fit . . . 12
gen_parnames . . . 13
genmixpars . . . 14
logfoldchange_fit . . . 14
logfoldchange_sim_fun . . . 16
logmean_fit . . . 17
logmean_sim_fun . . . 18
myrnormmix . . . 19
nllfun . . . 19
optimal.comp . . . 20
polyfun . . . 20
power_fun_ss . . . 21
read_data . . . 23
rnormmix0 . . . 24
sample_size_ss_interp . . . 24
ss_solver . . . 25
uniroot_ss . . . 26

Index **28**

`contour_plot_fun` *Contour plot for showing predicted power*

Description

Contour plot for showing predicted power

Usage

```
contour_plot_fun(
  combined_data,
  power_estimate,
  cont_breaks,
  multiple_samples = FALSE
)
```

Arguments

combined_data	data used for fitting gam
power_estimate	predicted power
cont_breaks	breaks for contour plot
multiple_samples	not currently used

Value

ggplot2 object

---

countdata_sim_fun *Simulate Count Data for Microbiome Studies*

Description

This function simulates count data for microbiome studies based on log mean, log fold change, and dispersion parameters. It supports generating data for multiple simulations and allows flexibility in specifying the number of control and treatment samples or samples per group.

Usage

```
countdata_sim_fun(
  logmean_param,
  logfoldchange_param,
  dispersion_param,
  nsamp_per_group = NULL,
  ncont = NULL,
  ntreat = NULL,
  notu,
  nsim = 1,
  disp_scale = 0.3,
  max_lfc = 15,
  maxlfc_iter = 1000,
  seed = NULL
)
```

Arguments

logmean_param	A list of parameters for simulating the log mean abundance.
logfoldchange_param	A list of parameters for simulating log fold change, containing: • par: Optimal parameters for log fold change fitting. • np: Number of components for the log fold change model. • sd_ord: Order of the polynomial for the standard deviation parameter.
dispersion_param	A list of dispersion parameters containing: • asymptDisp: Asymptotic dispersion parameter. • extraPois: Additional Poisson variation parameter.
nsamp_per_group	Number of samples per group (control and treatment). If provided, ncont and ntreat must not be specified.
ncont	Number of control samples. Specify along with ntreat when nsamp_per_group is not provided.
ntreat	Number of treatment samples. Specify along with ncont when nsamp_per_group is not provided.
notu	Number of operational taxonomic units (OTUs) to simulate.
nsim	Number of simulations to run. Default is 1.
disp_scale	Scale parameter for the dispersion. Default is 0.3.
max_lfc	Maximum allowable log fold change. Default is 15.
maxlfc_iter	Maximum number of iterations for ensuring log fold change is within max_lfc. Default is 1,000.
seed	Seed value for reproducibility. Default is NULL.

Value

A list containing:

- countdata_list: A list of count data matrices for each simulation.
- metadata_list: A list of metadata data frames for each simulation.
- logmean_list: A list of log mean vectors for each simulation.
- logfoldchange_list: A list of log fold change vectors for each simulation.
- treat_countdata_list: A list of treatment count data matrices for each simulation.
- control_countdata_list: A list of control count data matrices for each simulation.

Examples

```
# Load required packages
library(foreach)
library(doParallel)
# Define parameters
logmean_param <- list(mu = 0, sigma = 1)
```

```
logfoldchange_param <- list(par = rnorm(11), np = 2, sd_ord = 2)
dispersion_param <- list(asymptDisp = 0.1, extraPois = 0.05)

# Simulate count data
result <- countdata_sim_fun(
  logmean_param = logmean_param,
  logfoldchange_param = logfoldchange_param,
  dispersion_param = dispersion_param,
  nsamp_per_group = 10,
  notu = 50,
  nsim = 2,
  seed = 123
)

# Access simulation results
countdata <- result$countdata_list[[1]]
metadata <- result$metadata_list[[1]]
```

`deseq_fun_est` *Fold change and p-value estimations for simulations*

Description

Fold change and p-value estimations for simulations

Usage

```
deseq_fun_est(
  metadata_list,
  countdata_list,
  alpha_level = 0.1,
  group_colname,
  sample_colname,
  num_cores = 1,
  ref_name = NULL
)
```

Arguments

`metadata_list`	: list of metadata
`countdata_list`	: list of otu count data
`alpha_level`	The significance level for determining differential expression. Default is 0.1.
`group_colname`	column names of the groups or conditions
`sample_colname`	column names of the samples
`num_cores`	: number of cores
`ref_name`	reference level for fold change calculation. If NULL, the reference level is determined automatically — by default, the factor level that comes first is used as the reference.

Value

A list logfoldchange log fold change estimates

logmean is the log mean count for taxa (arithmetic mean for taxa across all subjects)

dispersion: dispersion estimates for each taxa

deseq_estimate is a dataframe containing results from deseq baseMean,log2FoldChange, lfcSE, pvalue, padj

normalised_count is the normalised count data

`deseqfun` *Estimate log fold changes using* `DESeq2`.

Description

This function estimates log fold changes (LFC) for microbiome count data using the DESeq2 package. It is strongly recommended to keep the following defaults:

- `minReplicatesForReplace=Inf`
- `cooksCutoff=TRUE`
- `independentFiltering=TRUE` These options are particularly useful when estimating fold changes to fit the mixture of Gaussian distributions.

Usage

```
deseqfun(
  countdata,
  metadata,
  alpha_level = 0.1,
  ref_name = NULL,
  group_colname,
  sample_colname,
  minReplicatesForReplace = Inf,
  cooksCutoff = TRUE,
  independentFiltering = TRUE,
  shrinkage_method = "normal"
)
```

Arguments

`countdata`	A matrix of OTU count data where rows represent taxa and columns represent samples.
`metadata`	A dataframe containing sample information with two rows: one for sample names and one for group names.
`alpha_level`	The significance level for determining differential expression. Default is 0.1.

`ref_name` reference level for fold change calculation. If NULL, the reference level is determined automatically — by default, the factor level that comes first is used as the reference.

`group_colname` column names of the groups or conditions

`sample_colname` column names of the samples

`minReplicatesForReplace` DESeq2's parameter to control the minimum number of replicates required for replacing outliers during dispersion estimation. Default is `Inf` (no replacement).

`cooksCutoff` DESeq2's parameter for removing outliers based on Cook's distance. Default is `TRUE` (outlier removal enabled).

`independentFiltering` DESeq2's parameter for independent filtering. Default is `TRUE`.

`shrinkage_method` DESeq2's shrinkage method for fold changes. Default is `"normal"`. Other options include `"apeglm"` or `"ashr"`.

Value

A list containing the following elements:

- `logfoldchange`: A vector of log fold change estimates for each taxa.
- `dispersion`: A vector of dispersion estimates for each taxa.
- `deseq_estimate`: A dataframe containing DESeq2 results, including `baseMean`, `log2FoldChange`, `lfcSE`, `pvalue`, and `padj`.
- `normalised_count`: A matrix of normalized count data.
- `dds`: DESeq object

Examples

```
# Example usage
set.seed(101)
nr = 10; nc = 35
countdata <- matrix(rpois(350, 3), ncol = nc, nrow = nr)
# Simulated OTU count data with 50 taxa and 10 samples
countdata  <- as.data.frame(countdata)
rownames(countdata)  <- paste("Sample", 1:nr, sep = "_")
colnames(countdata) <- paste("otu", 1:nc, sep = "_")
metadata <- data.frame(Samples = paste("Sample", 1:nr, sep = "_"),
             Groups = rep(c("Control", "Treatment"), each = 5))
sample_colname = "Samples"
group_colname  = "Groups"

result <- deseqfun(countdata, metadata, ref_name = "Control",
                   minReplicatesForReplace = Inf,
                   cooksCutoff = TRUE,
                   sample_colname = "Samples",
                   group_colname  = "Groups",
                   independentFiltering = TRUE,
```

```
                    shrinkage_method="normal")

# Examine the results
result$logfoldchange  # Log fold changes
result$logmean  # Log mean counts
result$dispersion  # Dispersion estimates
result$deseq_estimate  # DESeq2 results
result$normalised_count  # Normalized count data
```

`dispersion_fit` *Fit the non-linear function to dispersion estimates*

Description

Dispersion are estimated from the `DESeq2` package. The function fitted is of the form `a + b/(mean count)` where `a` represents the asymptotic dispersion level for high abundance taxa, and `b` captures additional dispersion variability.

Usage

```
dispersion_fit(dispersion, logmean)
```

Arguments

`dispersion`	dispersion estimates from deseq
`logmean`	vector of log mean abundance

Value

A list containing estimates for `a` and `b` and confidence intervals

Examples

```
logmean    =  rnorm(100)
dispersion =  abs(rnorm(100))
dispersion_fit(dispersion,logmean)
```

`dispersion_fun` *Calculate Dispersion for Microbiome Data*

Description

This function calculates the dispersion value for microbiome data based on the provided parameters: mean abundance, asymptotic dispersion, and extra Poisson dispersion.

Usage

```
dispersion_fun(mean_abund, asymptDisp, extraPois)
```

Arguments

`mean_abund`	Numeric value representing the mean abundance of the taxa.
`asymptDisp`	Numeric value for the asymptotic dispersion (the dispersion at high abundance).
`extraPois`	Numeric value for the extra Poisson dispersion (to model overdispersion).

Details

The dispersion is calculated using the formula:

$$dispersion = asymptDisp + \frac{extraPois}{meanAbund}$$

Value

A numeric value representing the dispersion.

Examples

```
mean_abund <- 10
asymptDisp <- 0.1
extraPois <- 0.05
dispersion_fun(mean_abund, asymptDisp, extraPois)
```

dnormmix *Density of a Normal Mixture Model*

Description

This function calculates the density of a normal mixture model for a given vector of parameters on the unconstrained scale (softmax(prob), mean, log(sd)).

Usage

```
dnormmix(x, par, logmean, ..., log = FALSE)
```

Arguments

x	Numeric vector of values at which to evaluate the density.
par	A vector of parameters on the unconstrained scale, including: • softmax(prob): Mixture probabilities (on the unconstrained scale, transformed via softmax). • mean: Means of the normal components. • log(sd): Logarithms of standard deviations of the normal components.
logmean	Numeric value representing the log of the mean parameter.
...	Additional arguments passed to the genmixpars function. Defaults: two components (np = 2) and a quadratic model for standard deviation parameters (sd_ord = 2).
log	Logical. If TRUE, the logarithm of the density is returned. Default is FALSE.

Value

A numeric vector of density values (or log-density values if log = TRUE) for the mixture model.

Examples

```
# Example parameters
x <- seq(-3, 3, length.out = 100)
## par <- c(-0.5, 0.5, log(0.8), log(1.2))  # Example: softmax probabilities, mean, log(sd)
set.seed(101); par <- rnorm(11)
logmean <- rep(0.1, length(x))  ## constant log mean

# Calculate density
density <- dnormmix(x, par, logmean)

# Calculate log-density
log_density <- dnormmix(x, par, logmean, log = TRUE)
```

dnormmix0 *Density function for the mixture of Gaussian distributions*

Description

takes pars as three vectors (prob, mean, sd), on constrained scale

Usage

```
dnormmix0(x, probs, muvals, sdvals, log = FALSE)
```

Arguments

x	vector
probs	mixture proportions
muvals	values of the mean
sdvals	values for standard deviataions
log	log scale

Value

likelihood

filter_low_count *Filter to remove low abundant taxa*

Description

Filter to retain only taxa with at least abund_thresh counts in at least sample_thresh samples

Usage

```
filter_low_count(
  countdata,
  metadata,
  abund_thresh = 5,
  sample_thresh = 3,
  sample_colname,
  group_colname
)
```

Arguments

countdata	otu table
metadata	dataframe with 2 rows sample names and group names
abund_thresh	minimum number of taxa abundance threshold
sample_thresh	minimum number of sample threshold
sample_colname	column names of the samples
group_colname	column names of the groups or conditions

Value

filtered otu count data

---

gam_fit *Title*

Description

Title

Usage

```
gam_fit(
  deseq_est_list,
  true_lfoldchange_list,
  true_lmean_list,
  grid_len = 50,
  alpha_level = 0.1
)
```

Arguments

deseq_est_list	a list containing fold change, pvalues and other estimates from DESeq2
true_lfoldchange_list	list containing simulated log fold change used for simulating the count data
true_lmean_list	list containing simulated log mean count used for simulating the count data
grid_len	number of grids for
alpha_level	significance level for power calculations

Value

A list fit_2d is the fitted scam object

power_estimate predicted power estimated using fit_2d

combined_data tibble containing pvlaues and used for GAM fit

gen_parnames *Generate Parameter Names for Mixture Model*

Description

This function generates parameter names for a Gaussian mixture model based on the number of components (np) and the order of the polynomial function (sd_ord) used to model the standard deviation parameters.

Usage

```
gen_parnames(np, sd_ord)
```

Arguments

np	Integer. The number of Gaussian components in the mixture model.
sd_ord	Integer. The order of the polynomial function used to model the standard deviation parameters. Possible values are: • 1: Linear function. • 2: Quadratic function.

Value

A character vector of parameter names, including:

- Logit-transformed probabilities (logitprob_1, ..., logitprob_(np-1)).
- Mean parameters (mu_int_1, mu_slope_1, ..., for each component).
- Log-transformed standard deviations (logsd_.1_1, logsd_.L_1, ..., depending on sd_ord and np).

Examples

```
# Generate parameter names for a 3-component mixture with linear standard deviation function
gen_parnames(np = 3, sd_ord = 1)

# Generate parameter names for a 4-component mixture with quadratic standard deviation function
gen_parnames(np = 4, sd_ord = 2)
```

`genmixpars` *generate normal mixture parameters (prob vector, mean vector, sd vector for a specified set of 'x' values (logmean)*

Description

generate normal mixture parameters (prob vector, mean vector, sd vector for a specified set of 'x' values (logmean)

Usage

```
genmixpars(x, pars, np = 2, sd_ord = 2)
```

Arguments

`x`	independent variable
`pars`	parameter vector: first logit-probs (np-1), then mean parameters (2 per component: intercepts, slopes), then var parameters (varord + 1 per component: intercepts, slopes, quad coeffs, etc.)
`np`	number of components in mixture
`sd_ord`	order of logsd model (2 = quadratic)

Value

A list

`probs: mixture proportions ()`

`muvals: mean values`

`sdvals: standard deviation values`

`logfoldchange_fit` *Fit a mixture of Gaussian distributions to log fold change*

Description

The standard deviation parameters are modeled either by linear or quadratic functions of log mean count and the mean parameter is modeled by linear functions of log mean count

Usage

```
logfoldchange_fit(
  logmean,
  logfoldchange,
  ncore = 2,
  max_sd_ord = 2,
  max_np = 5,
  minval = -5,
  maxval = 5,
  itermax = 100,
  NP = 800,
  seed = 100
)
```

Arguments

logmean	vector of log mean abundance
logfoldchange	vector of log fold change
ncore	number of cores to use
max_sd_ord	the maximum order of polynomial function to fit to standard deviation parameter. This must be either 1 (linear) or 2(quadratic)
max_np	maximum number of Gaussian components to check for
minval	minimum value for DEoptim search
maxval	maximum value for DEoptim search
itermax	maximum number of iterations
NP	the number of population members for DEoptim
seed	seed value

Value

A list.

```
par is a vector of the estimates of the mixture proportion,
the mean and standard deviation parameters,

np is the number of gaussian components fitted

sd_ord is the order for the function for the standard deviation

aic is the aic of the best fit
```

Examples

```
logmean       =  rnorm(100)
logfoldchange  =  rnorm(100)
logfoldchange_fit(logmean,logfoldchange)
```

`logfoldchange_sim_fun` *Simulate Log Fold Change Values*

Description

This function generates simulated log fold change (LFC) values based on the provided log mean abundance and LFC parameters. The simulation ensures that the generated LFC values remain within a specified maximum range by iterating until convergence or until a maximum iteration limit is reached.

Usage

```
logfoldchange_sim_fun(
  logmean_sim,
  logfoldchange_param,
  max_lfc = 15,
  max_iter = 10000,
  seed = 121
)
```

Arguments

`logmean_sim`	A numeric vector of simulated log mean abundances.
`logfoldchange_param`	A list containing parameters for the log fold change simulation: • `par`: Optimal parameters for the log fold change fit. • `np`: Optimal number of components for the log fold change model. • `sd_ord`: Order of the polynomial used for the standard deviation parameter of the log fold change.
`max_lfc`	A numeric value specifying the maximum allowable absolute log fold change value. Default is 15.
`max_iter`	An integer specifying the maximum number of iterations allowed to ensure all simulated LFC values are within the `max_lfc` range. Default is 10,000.
`seed`	random-number seed

Value

A numeric vector of simulated log fold change values (`lfc`).

Examples

```
set.seed(101)
# Define simulated log mean abundance
logmean_sim <- rnorm(100, mean = 0, sd = 1)

# Define parameters for log fold change simulation
```

```
logfoldchange_param <- list(
  par = rnorm(11),       # Example parameters
  np = 2,                # Number of components
  sd_ord = 2             # Order of polynomial for SD
)

# Simulate log fold change values
logfoldchange_sim_fun(
  logmean_sim = logmean_sim,
  logfoldchange_param = logfoldchange_param,
  max_lfc = 10,
  max_iter = 5000
)
```

`logmean_fit` *Fit a mixture of Gaussian Distributions to log mean count of taxa.*

Description

The optimal number of components to fit is chosen using parametric bootstrap method

Usage

```
logmean_fit(logmean, sig = 0.05, max.comp = 4, max.boot = 100, verb = FALSE)
```

Arguments

`logmean`	vector of log mean count of taxa
`sig`	significance level to compare against p-value to be used for parametric bootstrap calculation
`max.comp`	maximum number of Gaussian components to compare sequentially
`max.boot`	maximum number of bootstraps simulations
`verb`	If TRUE, it prints out updates of iterations of the algorithm

Value

A list containing the optimal number of Gaussian components fitted; and mean and variance parameter estimates from the fit

Examples

```
logmean  = rnorm(100)
logmean_fit(logmean,sig=0.05,max.comp=4,max.boot=100)
```

logmean_sim_fun *Simulate Log Means for OTUs*

Description

This function generates log means for a specified number of OTUs (Operational Taxonomic Units) based on the provided parameters. If a single mean is specified, the log means are drawn from a normal distribution. If multiple means and corresponding weights are specified, the log means are drawn from a mixture of normal distributions.

Usage

```
logmean_sim_fun(logmean_param, notu)
```

Arguments

logmean_param	A list containing the parameters for the distribution: • mu: A single value or a vector of mean(s) for the normal or mixture distribution. • sigma: The standard deviation(s) for the normal or mixture distribution. • lambda: (Optional) A vector of weights for the components of the mixture distribution. Required only if mu has more than one value.
notu	An integer specifying the number of OTUs to simulate.

Value

A numeric vector of simulated log means for the specified number of OTUs.

Examples

```
# Example 1: Single normal distribution
params_single <- list(mu = 0, sigma = 1)
logmean_sim_fun(logmean_param = params_single, notu = 100)

# Example 2: Mixture of normal distributions
params_mixture <- list(
  mu = c(-1, 1),
  sigma = c(0.5, 0.5),
  lambda = c(0.4, 0.6)
)
logmean_sim_fun(logmean_param = params_mixture, notu = 100)
```

myrnormmix *Simulating from a mixture of Gaussian*

Description

Simulating from a mixture of Gaussian

Usage

```
myrnormmix(par, logmean, ...)
```

Arguments

par	parameters (mean, standard deviation and mixture proportion of the mixture of Gaussian)
logmean	log mean count of taxa
...	other parameters taken by genmixpars

Value

random values from a a mixture of Gaussian

nllfun *Objective function*

Description

Objective function

Usage

```
nllfun(par, vals, logmean, np, sd_ord)
```

Arguments

par	parameters of the mixture of Gaussian
vals	values of log fold change
logmean	log mean count
np	number of Gaussian components
sd_ord	order of polynomial function to model standard deviation parameters (1 - linear function and 2- quad)

Value

value for the objective function

`optimal.comp` *Computes the optimal number of gaussian components for log mean count*

Description

The number of gaussian components is determined using the using parametric bootstrap

Usage

```
optimal.comp(logmean, sig = 0.05, max.comp = 4, max.boot = 100)
```

Arguments

`logmean`	vector of log mean abundances of taxa
`sig`	significance level to compare against p-value
`max.comp`	maximum number of Gaussian components to compare sequentially
`max.boot`	maximum number of bootstraps simulations

Value

The best number of components for fitting the distribution of log mean abundance

Examples

```
logmean  = rnorm(100)
optimal.comp(logmean,sig=0.05,max.comp=4,max.boot=100)
```

`polyfun` *General-purpose log-likelihood function, vectorized sum(pars*x^i)*

Description

General-purpose log-likelihood function, vectorized sum(pars*x^i)

Usage

```
polyfun(pars, x)
```

Arguments

`pars`	parameters
`x`	log mean count

Value

values representing output for the polynomial fuction (f(x))

Examples

```
polyfun(pars = c(1, 2, 3), x = 1:5)
polyfun(pars = c(1, 0, 3), x = 1)
```

`power_fun_ss` *Fit a smooth power model for sample size estimation*

Description

Fits a shape-constrained additive model (SCAM) to estimate statistical power as a function of mean abundance, absolute log fold change, and sample size. The function takes p-values obtained across simulation settings, converts them to rejection indicators based on a chosen significance level, and then models the probability of rejection using smooth terms.

Usage

```
power_fun_ss(
  pval_est_list,
  logmean_list,
  nsample_vec,
  logfoldchange_list,
  alpha_level = 0.1
)
```

Arguments

`pval_est_list`	A list of p-value vectors obtained from estimating the fold change estimates from the simulated datasets across different sample sizes. Each p-value corresponds to a test result for a particular combination of mean abundance, log fold change, and sample size.
`logmean_list`	A list containing the simulated log mean abundance values corresponding to the p-values in `pval_est_list`.
`nsample_vec`	A numeric vector of sample sizes used in the simulations.
`logfoldchange_list`	A list containing the simulated log fold change values corresponding to the p-values in `pval_est_list`.
`alpha_level`	Numeric significance level used to define rejection of the null hypothesis. Default is `0.1`.

Details

The returned model can be used as a smooth approximation to the empirical power surface, which is useful for interpolation and sample size prediction.

The function first pools all p-values from `pval_est_list` and converts them into binary rejection indicators:

$$I(p < \alpha)$$

where `alpha_level` is the chosen significance threshold.

A data frame is then constructed with:

- `logmean`: log mean abundance,
- `abs_lfc`: absolute log fold change,
- `pval_reject`: binary rejection indicator,
- `sample_size`: sample size,
- `logsample_size`: base-2 logarithm of sample size.

The function attempts to fit the following SCAM model:

$$\text{logit}\{P(\text{reject})\} = s(\text{logmean}, \text{abs\_lfc}, \text{bs = "tedmi"}) + s(\text{abs\_lfc}, \text{logsample\_size}, \text{bs = "tedmi"}) + s(\text{logmean}, \text{logsample\_siz}$$

using a binomial family.

If the full model fails due to numerical instability, a simpler fallback model is fitted instead:

$$\text{logit}\{P(\text{reject})\} = s(\text{logmean}, \text{abs\_lfc}, \text{bs = "tedmi"}) + s(\text{logsample\_size}, \text{bs = "mpi"})$$

A warning is issued when the fallback model is used.

Value

A list with two components:

- `combined_data`: A tibble containing the combined simulation inputs and rejection indicators used for model fitting.
- `gam_mod`: The fitted `scam` model object.

See Also

`scam::scam()`

Examples

```
#' @examples

# Example structure only
set.seed(101)
n = 70
pval_est_list = list(rnorm(n),rnorm(n))
logmean_list = list(rnorm(n),rnorm(n))
logfoldchange_list = list(rnorm(n),rnorm(n))
```

```
nsample_vec <- c(20, 40)
out <- power_fun_ss(
pval_est_list = pval_est_list,
logmean_list = logmean_list,
nsample_vec = nsample_vec,
logfoldchange_list = logfoldchange_list,
alpha_level = 0.1
)
out$combined_data
out$gam_mod
```

`read_data` *Extract specified data from a list of datasets*

Description

This function extracts a specific component (data) from a list of datasets. The component to extract is specified by the `extract_name` parameter, and the function returns a list containing the extracted data from each dataset.

Usage

```
read_data(dataset_list, extract_name)
```

Arguments

`dataset_list`	A list of datasets from which data will be extracted. Each element of the list is assumed to be a dataset (typically a list or dataframe).
`extract_name`	A string representing the name of the component or column to be extracted from each dataset in the `dataset_list`. The function looks for this name within each dataset.

Value

A list containing the extracted data. Each element corresponds to the extracted component from the datasets in `dataset_list`. The names of the list elements are taken from the names of `dataset_list`.

Examples

```
# Example dataset list
dataset1 <- list(countdata = matrix(1:9, nrow = 3), metadata = data.frame(id = 1:3))
dataset2 <- list(countdata = matrix(10:18, nrow = 3), metadata = data.frame(id = 4:6))
dataset_list <- list(dataset1 = dataset1, dataset2 = dataset2)

# Extract 'countdata' from each dataset in the list
result <- read_data(dataset_list, "countdata")
print(result)
```

rnormmix0 *general-purpose normal-mixture deviate generator: takes* matrices *of probabilities, means, sds*

Description

general-purpose normal-mixture deviate generator: takes *matrices* of probabilities, means, sds

Usage

```
rnormmix0(n, probs, muvals, sdvals)
```

Arguments

n	number of observations
probs	mixture proportions
muvals	values for the mean
sdvals	values for the standard deviation

Value

simulations

sample_size_ss_interp *Estimate sample size required to achieve a target statistical power*

Description

This function estimates the sample size required to achieve a specified target power using predictions from a fitted GAM/SCAM power model. The function evaluates predicted power across a grid of candidate sample sizes and identifies the smallest sample size for which the predicted power reaches or exceeds the target value. Linear interpolation is then used on the log2(sample size) scale to obtain a more precise estimate.

Usage

```
sample_size_ss_interp(
  target_power,
  logmean,
  abs_lfc,
  model,
  xmin = log2(5),
  xmax = log2(500),
  ngrid = 1000
)
```

Arguments

target_power	Numeric value specifying the desired statistical power.
logmean	Numeric value representing the log mean abundance.
abs_lfc	Numeric value representing the absolute log fold change.
model	A fitted GAM/SCAM model used to predict statistical power.
xmin	Numeric value giving the minimum log2(sample size) considered in the search. Default is log2(5).
xmax	Numeric value giving the maximum log2(sample size) considered in the search. Default is log2(500).
ngrid	Integer specifying the number of grid points used when searching for the sample size solution. Default is 1000.

Details

The function first constructs a grid of candidate sample sizes on the log2 scale between xmin and xmax. Predicted power values are then obtained from the fitted model for each grid point. The smallest sample size at which the predicted power reaches the target value is identified, and linear interpolation is used to refine the estimate.

Value

A numeric value representing the estimated sample size required to achieve the target power. Returns NA if the target power is not reached within the specified search range.

---

ss_solver *Solve for the sample size required to achieve a target statistical power*

Description

This function estimates the sample size required to achieve a specified target statistical power using a fitted GAM/SCAM power model. The function first attempts to solve for the sample size using a root-finding algorithm. If the root-finding procedure fails, a grid-based interpolation method is used as a fallback to obtain an approximate solution.

Usage

```
ss_solver(
  target_power,
  logmean,
  abs_lfc,
  model,
  xmin = log2(5),
  xmax = log2(500)
)
```

Arguments

target_power	Numeric value specifying the desired statistical power.
logmean	Numeric value representing the log mean abundance.
abs_lfc	Numeric value representing the absolute log fold change.
model	A fitted GAM/SCAM model used to predict statistical power.
xmin	Numeric value giving the minimum log2(sample size) considered in the search. Default is log2(5).
xmax	Numeric value giving the maximum log2(sample size) considered in the search. Default is log2(500).

Details

The function first attempts to compute the required sample size using a root-finding approach implemented in uniroot_ss(). If this method fails (for example, due to numerical issues or if the root cannot be bracketed within the specified interval), the function falls back to a grid-based interpolation approach implemented in sample_size_ss_interp().

A warning is issued if the target power is specified as 0 or 1, since these values correspond to unrealistic design targets in most practical applications.

Value

A numeric value representing the estimated sample size required to achieve the target power.

---

uniroot_ss *Sample Size estimation function using uniroot*

Description

Sample Size estimation function using uniroot

Usage

```
uniroot_ss(
  target_power,
  logmean,
  abs_lfc,
  model,
  xmin = log2(10),
  xmax = log2(5000),
  maxiter = 10000,
  max_report = 2000
)
```

Arguments

target_power	Numeric value specifying the desired statistical power.
logmean	Numeric value representing the log of the mean abundance.
abs_lfc	Numeric value representing the absolute log fold change.
model	A fitted GAM/SCAM model used to predict statistical power.
xmin	Numeric value giving the minimum sample size considered in the search.
xmax	Numeric value giving the maximum sample size considered in the search.
maxiter	maximum number of iterations
max_report	maximum group sample size to be predicted. Any predicted sample size that exceed max_report will be reported as "> max_report"

Value

A numeric value corresponding to the estimated sample size required to achieve the target power.

Index

contour_plot_fun, 2
countdata_sim_fun, 3

deseq_fun_est, 5
deseqfun, 6
dispersion_fit, 8
dispersion_fun, 9
dnormmix, 10
dnormmix0, 11

filter_low_count, 11

gam_fit, 12
gen_parnames, 13
genmixpars, 14

logfoldchange_fit, 14
logfoldchange_sim_fun, 16
logmean_fit, 17
logmean_sim_fun, 18

myrnormmix, 19

nllfun, 19

optimal.comp, 20

polyfun, 20
power_fun_ss, 21

read_data, 23
rnormmix0, 24

sample_size_ss_interp, 24
scam::scam(), *22*
ss_solver, 25

uniroot_ss, 26