Package 'QRscore'

June 5, 2026

Title Quantile Rank Score

Version 1.5.0

Date 2025-03-28

Description
In genomics, differential analysis enables the discovery of groups of genes implicating important biological processes such as cell differentiation and aging. Non-parametric tests of differential gene expression usually detect shifts in centrality (such as mean or median), and therefore suffer from diminished power against alternative hypotheses characterized by shifts in spread (such as variance). This package provides a flexible family of non-parametric two-sample tests and K-sample tests, which is based on theoretical work around non-parametric tests, spacing statistics and local asymptotic normality (Erdmann-Pham et al., 2022+ [arXiv:2008.06664v2]; Erdmann-Pham, 2023+ [arXiv:2209.14235v2]).

License GPL (>= 3)

Depends R (>= 4.4.0)

Imports MASS, pscl, arrangements, hitandrun, assertthat, dplyr, BiocParallel

Suggests devtools, DESeq2, knitr, rmarkdown, testthat, BiocStyle

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

NeedsCompilation no

VignetteBuilder BiocStyle

URL https://github.com/songlab-cal/QRscore

BugReports https://github.com/songlab-cal/QRscore/issues

biocViews StatisticalMethod, DifferentialExpression, GeneExpression, StructuralGenomics, GeneTarget

git_url https://git.bioconductor.org/packages/QRscore

git_branch devel

git_last_commit 8de552f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Author Fanding Zhou [cre, aut] (ORCID:
<https://orcid.org/0000-0003-1306-740X>),
Alan Aw [aut] (ORCID: <https://orcid.org/0000-0001-9455-7878>),
Dan Erdmann-Pham [aut],
Jonathan Fischer [aut] (ORCID: <https://orcid.org/0000-0002-1165-9930>),
Xurui Chen [ctb]

Maintainer Fanding Zhou <zhoufd@berkeley.edu>

Contents

`computeweight_disp` *Compute weights for dispersion shift analysis*

Description

Given the value of the estimated parameters for zero-inflated negative binomial distribution (β, μ, π), the sample sizes n and k (by default $n \geqslant k$), this function computes weights that target on changes of the dispersion parameter in zero-inflated negative binomial model.

Usage

```
computeweight_disp(beta, mu, pi, n, k, tail = 10^(-4), bigN = 10^6)
```

Arguments

`beta`	Target for number of successful trials, or dispersion parameter. Must be strictly positive.
`mu`	Non-negative mean of the uninflated negative binomial distribution.
`pi`	Zero inflation probability for structural zeros.
`n`	Sample size of y
`k`	Sample size of x
`tail`	Distribution tail trimmed for numerically calculate expectation. Default is 10^{-4}
`bigN`	Sampling numbers for numerically calculate expectation, this will only be applied when mean parameter is extremely large. Default is 10^6

Details

The function returns a list containing a weight vector and estimated variance for constructing test statistics.

Value

A list containing a weight vector (`$weight`) and estimated variance (`$var`) useful for constructing test statistics.

Examples

```
set.seed(1)
computeweight_disp(
    beta = 5, mu = 40, pi = 0.1, n = 100, k = 60,
    tail = 10^(-4), bigN = 10^6
)
```

---

computeweight_mean *Compute weights for mean shift analysis*

Description

Given the value of the estimated parameters for zero-inflated negative binomial distribution (β, μ, π),the sample sizes n and k (by default $n \geqslant k$), this function computes weights that target changes in the mean parameter of the zero-inflated negative binomial model.

Usage

```
computeweight_mean(beta, mu, pi, n, k)
```

Arguments

`beta`	Dispersion parameter of the negative binomial component. Must be strictly positive.
`mu`	Mean of the uninflated negative binomial distribution. Should be non-negative.
`pi`	Probability of zeros due to zero inflation.
`n`	Sample size of y
`k`	Sample size of x

Details

The function generates a vector of weights and an estimated variance that are used to construct test statistics. The weight vector is computed by considering how each potential count value from the combined distribution contributes to the expected shift in mean, considering both inflated and uninflated components.

Value

A list containing a weight vector (`$weight`) and estimated variance (`$var`) useful for constructing test statistics.

Examples

```
computeweight_mean(beta = 5, mu = 40, pi = 0.1, n = 100, k = 60)
```

`example_dataset` *Example Dataset for MOCHIS DEG Analysis*

Description

This dataset contains a subset of rounded normalized gene expression counts from GTEx whole blood tissue and associated labels for demonstration purposes.

Usage

```
example_dataset
```

Format

This dataset contains the following objects:

example_data1 A numeric matrix of normalized counts for all age groups.

labels1 A vector of sample labels for all age groups.

example_data2 A numeric matrix of normalized counts for age groups 20-39 and 60-79.

labels2 A vector of sample labels for age groups 20-39 and 60-79.

Source

Normalized counts from GTEx for package demonstration purposes.

Examples

```
data(example_dataset)
```

`example_dataset_raw_3000_genes`
Example Dataset for MOCHIS DEG Analysis with Raw Counts

Description

This dataset contains raw gene expression counts from 3000 genes in GTEx whole blood tissue for demonstration purposes.

Usage

```
example_dataset_raw_3000_genes
```

Format

This dataset contains a list with the following components

COUNTS A numeric matrix of raw counts for 3000 genes.

METADATA The metadata for the corresponding raw counts.

Source

Raw counts from GTEx for package demonstration purposes.

Examples

```
data(example_dataset_raw_3000_genes)
```

`getCompositionPValue` *Approximate p-value by Resampling Integer Compositions*

Description

Given the value of the test statistic t, the sample sizes n and k,power exponent p and vector of weights that together determine the test statistic (by default $n \geqslant k$), as well as the user-specified resampling number (by default this is 5000), performs resampling from the collection of integer compositions to approximate the p-value of the observed test statistic.

Usage

```
getCompositionPValue(t, n, k, p, wList, alternative, type, resamp_number)
```

Arguments

`t`	Value of test statistic $\|\|S_{n,k}(D)/n\|\|^p_{p,\boldsymbol{w}}$ computed from data D
`n`	Sample size of y
`k`	Sample size of x
`p`	Power exponent of test statistic
`wList`	Weight vector
`alternative`	Character string that should be one of `"two.sided"` (default), `"greater"` or `"less"`
`type`	If using resampling approximation, either an unbiased estimate of (`"unbiased"`, default), or valid, but biased estimate of, (`"valid"`) p-value (see Hemerik and Goeman, 2018), or both (`"both"`). Default is `"unbiased"`.
`resamp_number`	Number of compositions of n to draw (default is 5000)

Details

For n and k small enough ($n \leqslant 40, k \leqslant 10$), computes exact p-value by computing the test statistic across all k-compositions of n under the uniform distribution.

The function returns a two-sided p-value by default, which is more conservative. Users can choose other p-values corresponding to different alternatives; see documentation on `alternative`. Note that the interpretation of the choice of `alternative` depends on the choice of weight vector. For example, a weight vector that is a quadratic kernel will upweight the extreme components of the weight vector. For this choice, setting `alternative` to be `greater` translates into an alternative hypothesis of a bigger spread in the larger sample (the one with sample size n).

Dependencies: arrangements::compositions

Value

p-value (scalar)

Examples

```
getCompositionPValue(
    t = 0.5, n = 50, k = 11, p = 1, wList = (10:0) / 10,
    alternative = "two.sided", type = "unbiased", resamp_number = 5000
)
```

`QRscore` *QRscore: Nonparametric Quantile Rank Score Tests for Distributional Shifts in Gene Expression*

Description

The QRscore package implements a family of nonparametric two-sample, and multi-sample tests for detecting shifts in central tendency (mean) and spread (variance/dispersion) in gene expression data. The package includes functionality for negative binomial and zero-inflated negative binomial models, resampling-based and asymptotic approximations, and is designed to be robust, flexible and efficient.

Main Functions

- `QRscoreTest()`: Wrapper for one-, two-, and multi-sample QRscore tests.
- `QRscoreGenetest()`: DE pipeline over many genes.

Author(s)

Maintainer: Fanding Zhou <zhoufd@berkeley.edu> (ORCID)

Authors:

- Alan Aw <nalawanij@gmail.com> (ORCID)
- Dan Erdmann-Pham <erdpham@stanford.edu>
- Jonathan Fischer <jfischer1@ufl.edu> (ORCID)

Other contributors:

- Xurui Chen <xuruichen@berkeley.edu> [contributor]

See Also

Useful links:

- https://github.com/songlab-cal/QRscore
- Report bugs at https://github.com/songlab-cal/QRscore/issues

---

`QRscoreGenetest` *Perform Differential Expression Analysis using QRscore*

Description

This function performs differential expression analysis using the QRscore method.

Usage

```
QRscoreGenetest(
  expr_matrix,
  labels,
  size_factors = NULL,
  pairwise_test = FALSE,
  pairwise_logFC = FALSE,
  test_mean = TRUE,
  test_dispersion = FALSE,
  num_cores = 1,
  verbose = FALSE,
  rank = TRUE,
  seed = 1,
  ...
)
```

Arguments

`expr_matrix`	A prefiltered raw count matrix (gene $\times$ sample).
`labels`	A vector of labels corresponding to the samples.
`size_factors`	A vector of size factors estimated from DESeq2
`pairwise_test`	Logical, whether to perform pairwise test statistics for more than two groups.
`pairwise_logFC`	Logical, whether to calculate pairwise log fold changes for mean and variance.
`test_mean`	Logical, whether to test the mean.
`test_dispersion`	Logical, whether to test the dispersion.
`num_cores`	Integer, number of cores to use for parallel processing.
`verbose`	Logical, whether to suppress all messages except warnings and errors.
`rank`	Logical. If TRUE, returns a ranked list of DEGs or DDGs sorted by p-values.
`seed`	Integer. Set seed in parallel computing.
`...`	Additional arguments passed to the `QRscoreTest` function.

Value

A list with two data frames: `mean_test` and `var_test`.

Examples

```
data(example_dataset)
results <- QRscoreGenetest(
  expr_matrix = example_dataset$example_data1,
  labels = example_dataset$labels1,
  size_factors = example_dataset$size_factors1,
  pairwise_test = FALSE, pairwise_logFC = TRUE,
  test_mean = TRUE, test_dispersion = TRUE, num_cores = 2,
  approx = "asymptotic"
)
head(results$var_test)

results2 <- QRscoreGenetest(
  expr_matrix = example_dataset$example_data2,
  labels = example_dataset$labels2,
  size_factors = example_dataset$size_factors2,
  pairwise_test = TRUE, pairwise_logFC = TRUE,
  test_mean = TRUE, test_dispersion = FALSE, num_cores = 2,
  approx = "asymptotic"
)
head(results2$mean_test)
```

QRscoreTest *QRscore Test*

Description

This function performs statistical tests on data from one or more groups using summary statistics computed on weighted linear combinations of powers of spacing statistics. It is capable of conducting one-sample tests, two-sample tests, and multi-sample tests, utilizing either user-defined weights or automatically generated weights based on Negative Binomial (NB) or Zero-Inflated Negative Binomial (ZINB) models.

Usage

```
QRscoreTest(
  samples,
  labels = NULL,
  size_factors = NULL,
  p = NULL,
  wList = NULL,
  alternative = "two.sided",
  approx = "resample",
  type = "unbiased",
  n_mom = 100,
  resamp_number = 5000,
  zero_inflation = TRUE,
  LR.test = FALSE,
```

```
  pi_threshold = 0.95,
  gene.name = NULL,
  measure = "mean",
  perturb = TRUE,
  use_base_r = TRUE
)
```

Arguments

samples	A numeric vector containing all sample measurements.
labels	An optional vector of group labels corresponding to each entry in samples.
size_factors	Optional vector of size factors for weight estimation. If provided, log-transformed size factors are used as offsets in NB/ZINB model fitting.
p	The exponent used in the power sum test statistic, required if wList is not NULL.
wList	An optional vector of weights; if NULL, weights are estimated using an NB or ZINB model for multiple groups.
alternative	Specifies the alternative hypothesis; must be one of "two.sided", "greater", or "less".
approx	The method used for p-value approximation, either "resample" or "asymptotic".
type	Specifies if the estimation of the p-value should be "unbiased", "valid", or "both".
n_mom	The number of moments to accompany the approximation, relevant for moment-based methods.
resamp_number	The number of resampling iterations used if approx is "resample".
zero_inflation	Indicates whether to account for zero inflation in model-based weight estimation.
LR.test	Whether a likelihood ratio test is used to decide between NB and ZINB models.
pi_threshold	Threshold for the proportion of zeros in ZINB models; results in NA if exceeded.
gene.name	Optional identifier for a gene, used in output messages.
measure	Specifies the statistical measure to be analyzed ("mean" or "dispersion") when weights are auto-generated.
perturb	Boolean to indicate if data should be perturbed slightly to prevent ties.
use_base_r	Boolean to decide whether to use base R functions for certain edge cases like Mann-Whitney tests.

Details

For one-sample tests, the function assesses the uniformity of data entries. For two-sample and multi-sample tests, it evaluates whether groups are drawn from the same distribution, with alternative hypotheses considering differences in means or dispersions.

If the weights and p are given, the function calculates the test statistic as:

$$||S_k||_{p,\boldsymbol{w}}^p = \sum_{j=1}^{k} w_i S_k[j]^p$$

where w_i are weights, x_i are data points, and p is the power specified.

In two-sample and multi-sample settings without specified weights, the function can automatically estimate weights using score function for a Negative Binomial or a Zero-Inflated Negative Binomial model, optimizing for dispersion or mean shifts.

Value

Returns the p-value of the test.

Examples

```
set.seed(1)
# One-sample test example with normally distributed data
data <- abs(rnorm(10))
QRscoreTest(data,
    p = 2, wList = rep(1, 10), alternative = "two.sided",
    approx = "resample"
)

# Two-sample test with specified weights using normally distributed data
group1 <- rnorm(120, sd = 1)
group2 <- rnorm(120, sd = 2) # Different mean
data <- c(group1, group2)
labels <- c(rep("Group1", 120), rep("Group2", 120))
QRscoreTest(
    samples = data, labels = labels, p = 1, wList = c(60:0, seq_len(60)),
    alternative = "two.sided", approx = "resample"
)

# Two-sample test with automatically estimated weights from NB model
group1 <- rzinbinom(120, size = 2, mu = 20, pi = 0)
group2 <- rzinbinom(100, size = 2, mu = 30, pi = 0) # Different mean
data <- c(group1, group2)
labels <- c(rep("Group1", 120), rep("Group2", 100))
QRscoreTest(
    samples = data, labels = labels,
    approx = "asymptotic", measure = "mean", zero_inflation = FALSE
)

# Two-sample test with automatically estimated weights from ZINB model
group1 <- rzinbinom(100, size = 2, mu = 40, pi = 0.1)
group2 <- rzinbinom(200, size = 1, mu = 40, pi = 0.1)
data <- c(group1, group2)
labels <- c(rep("Group1", 100), rep("Group2", 200))
QRscoreTest(
    samples = data, labels = labels, alternative = "two.sided",
    approx = "asymptotic", measure = "dispersion"
)

# Three-sample test with automatically estimated weights from NB model
group1 <- rzinbinom(150, size = 1, mu = 30, pi = 0.1)
group2 <- rzinbinom(100, size = 2, mu = 30, pi = 0.1)
group3 <- rzinbinom(30, size = 3, mu = 30, pi = 0.1)
data <- c(group1, group2, group3)
labels <- c(rep("Group1", 150), rep("Group2", 100), rep("Group3", 30))
QRscoreTest(
    samples = data, labels = labels, alternative = "two.sided",
    approx = "asymptotic", measure = "dispersion"
)

# Two-sample NB test with size factors
sf <- runif(200, 0.5, 1.5)
```

```
x <- rnbinom(100, size = 2, mu = sf[1:100] * 20)
y <- rnbinom(100, size = 2, mu = sf[101:200] * 35)
QRscoreTest(
    samples = c(x, y), labels = rep(c("A", "B"), each = 100),
    size_factors = sf, measure = "mean", zero_inflation = FALSE
)
```

`QRscore_Flex` *Flexible Non-Parametric One- and Two-Sample Tests (Native R version)*

Description

Given data consisting of either a single sample $\boldsymbol{x} = (x_1, \ldots, x_k)$, or two samples $\boldsymbol{x} = (x_1, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, this function uses summary statistics computed on weighted linear combinations of powers of the spacing statistics S_k (former) or $S_{n,k}$ (latter).

Usage

```
QRscore_Flex(
  x,
  y = NULL,
  p = 1,
  wList,
  alternative = "two.sided",
  approx = "resample",
  type = "unbiased",
  n_mom = NULL,
  resamp_number = 5000
)
```

Arguments

`x`	First sample
`y`	Second sample
`p`	Exponent value in defining test statistic (must be integer)
`wList`	Vector of weights. It should have length equal to x when y is `NULL`, and one more than the length of x when y is not `NULL`
`alternative`	How p-value should be computed; i.e., a character specifying the alternative hypothesis, must be one of `"two.sided"`, `"greater"` or `"less"`
`approx`	Which approximation method to use (choose `"resample"`, `"asymptotic"`)
`type`	If using resampling approximation, either an unbiased estimate of (`"unbiased"`, default), or valid, but biased estimate of, (`"valid"`) p-value (see Hemerik and Goeman, 2018), or both (`"both"`). Default is `"unbiased"`.
`n_mom`	The number of moments to accompany the approximation (recommended 200, if not at least 100)
`resamp_number`	Number of k-compositions of n or simplex vectors in $[0, 1]^k$ to draw

Details

More precisely, this function does the following:

For a single sample x, the function tests for uniformity of its entries. When $p = 2$ and a particular choice of $\boldsymbol{w}$ is specified, we recover Greenwood's test.

For two samples, the function tests the null of $\boldsymbol{x}$ and $\boldsymbol{y}$ being drawn from the same distribution (i.e., stochastic equality), against flexible alternatives that correspond to specific choices of the test statistic parameters, $\boldsymbol{w}$ (weight vector) and p (power). These parameters not only determine the test statistic $||S_k||^p_{p,\boldsymbol{w}} = \sum_{j=1}^{k} w_i S_k[j]^p$ (analogously defined for $||S_{n,k}||^p_{p,\boldsymbol{w}}$), but also encode alternative hypotheses ranging from different populational means (i.e., $\mu_x \neq \mu_y$), different populational spreads (i.e., $\sigma_x^2 \neq \sigma_y^2$), etc.

Additional tuning parameters include: (1) choice of p-value computation (one- or two-sided); (2) approximation method (3) number of moments accompanying the approximation chosen if using moment-based approximation (recommended 200, typically at least 100); and (4) in case of two samples, whether the user prefers to use exact discrete moments (more accurate but slower) or to use continuous approximations of the discrete moments (less accurate but faster).

Currently, only resampling and Gaussian asymptotics are supported. Both are efficient and well-calibrated. For $n \geqslant 100$ and $k \geqslant 50$ such that $\frac{k}{n} \geqslant 0.001$, function automatically uses Gaussian approximation to the null.

Dependencies: functions in `utils.R`

Value

Returns the p-value.

Examples

```
set.seed(1)
# One-sample examples
QRscore_Flex(
    x = abs(rnorm(10)), p = 2, wList = rep(1, 10),
    alternative = "two.sided", approx = "resample"
)

# Two-sample examples
QRscore_Flex(
    x = abs(rnorm(30)), y = abs(rnorm(100)), p = 2,
    wList = rep(1, 31), alternative = "two.sided",
    approx = "resample", resamp_number = 5000
)

QRscore_Flex(
    x = abs(rnorm(100)), y = abs(rnorm(100)), p = 1,
    wList = 0:100, alternative = "two.sided",
    approx = "asymptotic"
)
```

QRscore_ZINB *Non-Parametric Two-Sample Tests Designed for Testing Differences in Mean or Dispersion Parameters in (Zero-Inflated) Negative Binomial Distributions.*

Description

This function evaluates the null hypothesis that two samples, $\boldsymbol{x}$ and $\boldsymbol{y}$, are drawn from the same distribution, specifically designed for NB or ZINB models. It is particularly effective in detecting shifts in either the mean or the dispersion parameters.

Usage

```
QRscore_ZINB(
  x,
  y,
  size_factors = NULL,
  zero_inflation = TRUE,
  LR.test = FALSE,
  approx = "resample",
  alternative = "two.sided",
  resamp_num = 20000,
  pi_threshold = 0.95,
  gene.name = NULL,
  measure = "mean",
  p_value = TRUE
)
```

Arguments

x	First sample
y	Second sample
size_factors	Optional vector of size factors for weight estimation. If provided, log-transformed size factors are used as offsets in NB/ZINB model fitting.
zero_inflation	If TRUE, automatically chooses between ZINB and NB models based on the data; if FALSE, applies NB model estimation.
LR.test	Whether to use a likelihood ratio test to determine which model (NB or ZINB) to fit
approx	Which approximation method to use (default `resample`)
alternative	How p-value should be computed; must be one of "`two.sided`", "`greater`" or "`less`".
resamp_num	Number of k-compositions of n or simplex vectors in $[0,1]^k$ to draw
pi_threshold	Threshold for estimated proportion of zeros in ZINB model
gene.name	Optional, name of the gene if applicable, used for customized messages.
measure	Specifies whether to test for shifts in "`mean`" or "`dispersion`".
p_value	If TRUE, returns a p-value, else returns test statistics and weights.

Details

The function automatically computes optimal weights for the chosen model and derives a p-value based on the selected test statistic and approximation method.

Additional tuning parameters include: (1) whether to use a likelihood ratio test to determine which model (NB or ZINB) to fit, (2) the approximation method (default is resampling, with asymptotic estimation for large samples),(3) choice of p-value computation (one- or two-sided), (4) threshold for estimated proportion of zeros in ZINB model (returns NA if exceeded).

Dependencies: pscl::zeroinfl, MASS::glm.nb, and auxiliary functions from `auxiliary.R`

Value

p-value or test statistics depending on `p_value` parameter.

Examples

```
set.seed(1)
# Two-sample example comparing mean shifts
QRscore_ZINB(
    x = rzinbinom(100, size = 2, mu = 20, pi = 0),
    y = rzinbinom(100, size = 2, mu = 30, pi = 0),
    zero_inflation = FALSE, LR.test = FALSE, alternative = "greater",
    approx = "asymptotic", measure = "mean"
)

# Two-sample example comparing dispersion shifts
QRscore_ZINB(
    x = rzinbinom(100, size = 2, mu = 20, pi = 0.1),
    y = rzinbinom(100, size = 1, mu = 20, pi = 0.1),
    zero_inflation = TRUE, LR.test = TRUE, alternative = "two.sided",
    approx = "asymptotic", measure = "dispersion"
)

# Two-sample example with significant zero inflation and variance shift
QRscore_ZINB(
    x = rzinbinom(30, size = 4, mu = 20, pi = 0.1),
    y = rzinbinom(30, size = 1, mu = 20, pi = 0.3),
    zero_inflation = TRUE, LR.test = FALSE, alternative = "two.sided",
    approx = "resample", resamp_num = 50000, measure = "dispersion"
)
# Two-sample example with size factors, zero inflation, and dispersion shift
sf_x <- runif(100, 0.5, 1.5)
sf_y <- runif(100, 0.5, 1.5)
QRscore_ZINB(
    x = rzinbinom(100, size = 4, mu = sf_x * 25, pi = 0.1),
    y = rzinbinom(100, size = 1,   mu = sf_y * 25, pi = 0.1),
    size_factors = c(sf_x, sf_y),
    zero_inflation = TRUE, LR.test = TRUE, alternative = "two.sided",
    approx = "resample", resamp_num = 20000, measure = "dispersion"
)
```

`QRscore_ZINB_nSamples` *Multi-Sample Nonparametric Test for Mean or Dispersion Differences in (Zero-Inflated) Negative Binomial Distributions.*

Description

This function conducts statistical tests across multiple samples to evaluate the null hypothesis that all groups are drawn from the same distribution. It is optimized for data modeled by Negative Binomial (NB) or Zero-Inflated Negative Binomial (ZINB) distributions and is capable of detecting shifts in mean or dispersion parameters. The function can handle any number of groups and automatically computes optimal weights for the specified measure (mean or dispersion).

Usage

```
QRscore_ZINB_nSamples(
  samples,
  labels,
  size_factors = NULL,
  zero_inflation = TRUE,
  LR.test = FALSE,
  approx = "resample",
  resamp_num = 20000,
  pi_threshold = 0.95,
  gene.name = NULL,
  measure = "mean",
  perturb = TRUE
)
```

Arguments

`samples`	Vector of all sample measurements.
`labels`	Group labels for each sample.
`size_factors`	Optional vector of size factors for weight estimation. If provided, log-transformed size factors are used as offsets in NB/ZINB model fitting.
`zero_inflation`	Boolean, if TRUE, the function chooses between ZINB and NB models based on data; if FALSE, only NB model is applied.
`LR.test`	Boolean, if TRUE, performs a likelihood ratio test to select between NB and ZINB models.
`approx`	The method used for p-value approximation; "resample" (default) or "asymptotic".
`resamp_num`	The number of resampling iterations used if `approx` is "resample".
`pi_threshold`	Threshold for proportion of zeros at which to return NA, indicating unreliable results due to excessive zero inflation.
`gene.name`	Optional, name of the gene if applicable, enhancing the relevance of output in genetic studies.
`measure`	Specifies whether the test focuses on "mean" or "dispersion" differences.
`perturb`	Boolean, if TRUE, adds small noise to data to avoid ties and improve model stability.

Details

The computation involves constructing a B matrix that transforms group-specific scores into a space where independence among groups is maximized. It then uses these transformed scores to calculate a test statistic, which follows a chi-square distribution under the null hypothesis.

Additional tuning parameters allow customization of the model fitting and statistical testing, including:

- Selection between NB and ZINB models based on presence of zero inflation.
- Choice of approximation method for computing p-values - 'asymptotic' is recommended.
- Decision criteria for statistical tests (one-sided or two-sided).
- Threshold for the estimated proportion of zeros beyond which results are considered unreliable.

Dependencies: Requires `pscl::zeroinfl` for zero-inflated models, `MASS::glm.nb` for NB models, and other auxiliary functions as needed.

Value

Returns the p-value of the test if `p_value` is TRUE, otherwise returns test statistics and weights.

Examples

```
set.seed(1)
data <- c(
    rnbinom(100, size = 2, mu = 20), rnbinom(100, size = 2, mu = 25),
    rnbinom(100, size = 2, mu = 30)
)
labels <- factor(c(rep("Group1", 100), rep("Group2", 100),
          rep("Group3", 100)))
QRscore_ZINB_nSamples(
    samples = data, labels = labels,
    zero_inflation = FALSE, LR.test = FALSE, approx = "resample",
    resamp_num = 5000, pi_threshold = 0.95, measure = "mean"
)

# Example with zero inflation and dispersion shift detection
data_zi <- c(
    rzinbinom(100, size = 2, mu = 20, pi = 0.1),
    rzinbinom(100, size = 3, mu = 20, pi = 0.1),
    rzinbinom(100, size = 4, mu = 20, pi = 0.1)
)
labels_zi <- factor(c(rep("Group1", 100), rep("Group2", 100),
              rep("Group3", 100)))
QRscore_ZINB_nSamples(
    samples = data_zi, labels = labels_zi,
    zero_inflation = TRUE, LR.test = TRUE, approx = "asymptotic",
    resamp_num = 2000, pi_threshold = 0.95, measure = "dispersion"
)
# Multi-sample NB example with size factors
sf <- runif(300, 0.5, 1.5)
data_sf <- c(
    rnbinom(100, size = 2, mu = sf[1:100] * 20),
    rnbinom(100, size = 2, mu = sf[101:200] * 30),
    rnbinom(100, size = 2, mu = sf[201:300] * 40)
)
```

```
labels_sf <- factor(rep(c("Group1", "Group2", "Group3"), each = 100))
QRscore_ZINB_nSamples(
    samples = data_sf, labels = labels_sf, size_factors = sf,
    zero_inflation = FALSE, LR.test = FALSE, approx = "asymptotic",
    resamp_num = 10000, pi_threshold = 0.95, measure = "mean"
)
```

---

rank_x *Retrieve indices of x_i's after merging $\boldsymbol{x}$ and $\boldsymbol{y}$ in ascending order.*

Description

Given data consisting of either a single sample $\boldsymbol{x} = (x_1, \ldots, x_k)$, or two samples $\boldsymbol{x} = (x_1, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, this function obtains the indices of x_i's after merging $\boldsymbol{x}$ and $\boldsymbol{y}$ in ascending order.

Usage

```
rank_x(x, y = NULL, ties.break = TRUE)
```

Arguments

x	First sample
y	Second sample
ties.break	Whether to break the ties when ordering x and y. Default is 'TRUE'.

Details

Dependencies: None

Value

Ranks of x_i's after merging x and y in ascending order

Examples

```
set.seed(1)
rank_x(x = abs(rnorm(10)))
rank_x(x = abs(rnorm(10)), y = abs(rnorm(100)))
rank_x(
    x = rnbinom(10, size = 5, prob = 0.3),
    y = rnbinom(20, size = 2, prob = 0.3), ties.break = TRUE
)
```

`rzinbinom` *Generate random samples from a zero-inflated negative binomial distribution*

Description

Generate random samples from a zero-inflated negative binomial distribution

Usage

```
rzinbinom(n, mu, theta, size, pi)
```

Arguments

`n`	Integer. Number of samples to generate.
`mu`	Numeric. Mean of the negative binomial distribution.
`theta`	Numeric. Dispersion parameter (size) of the negative binomial distribution.
`size`	Numeric. Alternative name for the dispersion parameter (used interchangeably with `theta`).
`pi`	Numeric. Zero-inflation probability; must be in the range $[0, 1]$.

Value

A vector of random samples from a Zero-Inflated Negative Binomial (ZINB) distribution.

Examples

```
set.seed(1)
rzinbinom(n = 5, mu = 10, theta = 5, pi = 0.2)
```

Index