

# Package ‘TailRank’

May 7, 2026

**Title** The Tail-Rank Statistic

**Version** 3.2.4

**Date** 2025-04-07

**Description** Implements the tail-rank statistic for selecting biomarkers from a microarray data set, an efficient nonparametric test focused on the distributional tails. See <https://gitlab.com/krcoombes/coombeslab/-/blob/master/doc/papers/tolstoy-new.pdf>.

**Depends** R (>= 3.0), oompaBase (>= 3.0.1)

**Imports** methods, graphics, stats, Biobase, oompaData

**Suggests** xtable

**License** Apache License (== 2.0)

**LazyLoad** yes

**URL** <http://oompa.r-forge.r-project.org/>

**NeedsCompilation** no

**Author** Kevin R. Coombes [aut, cre]

**Maintainer** Kevin R. Coombes <krc@silicovore.com>

**Repository** CRAN

**Date/Publication** 2025-04-07 12:30:02 UTC

## Contents

BetaBinomial . . . . .	2
biomarkerPowerTable . . . . .	3
BMPT-class . . . . .	5
tailRankPower . . . . .	6
TailRankTest . . . . .	8
TailRankTest-class . . . . .	10
TailRankTest-methods . . . . .	12
toleranceBound . . . . .	13

<b>Index</b>	<b>16</b>
--------------	-----------

**Description**

Density, distribution function, quantile function, and random generation for the beta-binomial distribution. A variable with a beta-binomial distribution is distributed as binomial distribution with parameters  $N$  and  $p$ , where the probability  $p$  of success itself has a beta distribution with parameters  $u$  and  $v$ .

**Usage**

```
dbb(x, N, u, v, log = FALSE)
pbb(q, N, u, v)
qbb(p, N, u, v)
rbb(n, N, u, v)
```

**Arguments**

<code>x</code>	vector of quantiles
<code>q</code>	vector of quantiles
<code>p</code>	vector of probabilities
<code>n</code>	number of observations
<code>N</code>	number of trials ( a positive integer)
<code>u</code>	first positive parameter of the beta distribution
<code>v</code>	second positive parameter of the beta distribution
<code>log</code>	A logical value; if true, values are returned on the log scale

**Details**

The beta-binomial distribution with parameters  $N$ ,  $u$ , and  $v$  has density given by

$$\text{choose}(N, x) * \text{Beta}(x + u, N - x + v) / \text{Beta}(u, v)$$

for  $u > 0$ ,  $v > 0$ , a positive integer  $N$ , and any nonnegative integer  $x$ . Although one can express the integral in closed form using generalized hypergeometric functions, the implementation of distribution function used here simply relies on the the cumulative sum of the density.

The mean and variance of the beta-binomial distribution can be computed explicitly as

$$\mu = \frac{Nu}{u + v}$$

and

$$\sigma^2 = \frac{Nuv(N + u + v)}{(u + v)^2(1 + u + v)}$$

**Value**

dbb gives the density, pbb gives the distribution function, qbb gives the quantile function, and rbb generates random deviates.

**Author(s)**

Kevin R. Coombes <krc@silicovore.com>

**See Also**

[dbeta](#) for the beta distribution and [dbinom](#) for the binomial distribution.

**Examples**

```
# set up parameters
w <- 10
u <- 0.3*w
v <- 0.7*w
N <- 12
# generate random values from the beta-binomial
x <- rbb(1000, N, u, v)

# check that the empirical summary matches the theoretical one
summary(x)
qbb(c(0.25, 0.50, 0.75), N, u, v)

# check that the empirical histogram matches the theoretical density
hist(x, breaks=seq(-0.5, N + 0.55), prob=TRUE)
lines(0:N, dbb(0:N, N, u,v), type='b')
```

---

biomarkerPowerTable *Power tables for the tail-rank test*

---

**Description**

Compute an array of power tables for the tail-rank.test.

**Usage**

```
biomarkerPowerTable(G, N1=20, N2=seq(25, 250, by=25),
  psi = c(0.95, 0.99), conf=0.99,
  phi = seq(0.10, 0.50, by = 0.05), model="bb")
```

**Arguments**

G	An integer; the number of genes being assessed as potential biomarkers. Statistically, the number of hypotheses being tested.
N1	An integer; the number of "train" or "healthy" samples used.
N2	An integer; the number of "test" or "cancer" samples used.
psi	A real number between 0 and 1; the desired specificity of the test.
conf	A real number between 0 and 1; the confidence level of the results. Can be obtained by subtracting the family-wise Type I error from 1.
phi	A real number between 0 and 1; the sensitivity that one would like to be able to detect, conditional on the specificity.
model	A character string that determines whether power and significance are computed from abinomial or a beta-binomial (bb) model.

**Value**

Returns a list of objects of the [BMPT](#) class. Each item in the list consists of a two-dimensional table (indexed by the sample sizes N and the sensitivities phi) with scalars recording the values of G, conf, and psi that were used to generate it.

**Note**

Default values of the optional arguments (N, psi, conf, phi) are included in the usage examples.

**Author(s)**

Kevin R. Coombes <krc@silicovore.com>

**See Also**

[TailRankTest](#), [tailRankPower](#), [biomarkerPowerTable](#), [matrixMean](#), [toleranceBound](#)

**Examples**

```
stuff <- biomarkerPowerTable(10000, 20,
                             c(10, 20, 50, 100, 250, 500),
                             c(0.95, 0.99),
                             c(0.99, 0.95),
                             seq(0.1, 0.7, by=0.1))
lapply(stuff, summary)
```

**Description**

A class for producing BioMarker Power Tables (BMPT), and methods for accessing them. This class is primarily an implementation detail for the function [biomarkerPowerTable](#).

**Usage**

```
BMPT(G, psi, conf, power)
## S4 method for signature 'BMPT'
print(x,...)
## S4 method for signature 'BMPT'
summary(object,...)
```

**Arguments**

G	A positive integer.
psi	A real number between 0 and 1.
conf	A real number between 0 and 1.
power	A data frame.
x	A BMPT object.
object	A BMPT object.
...	Extra graphical parameters

**Creating objects**

Although objects can be created using `new`, the preferred method is to use the constructor function `BMPT`. In practice, these objects are most likely to be created using the more general interface through [biomarkerPowerTable](#).

**Slots**

**G:** A positive integer; the number of genes being assessed as potential biomarkers. Statistically, the number of hypotheses being tested.

**psi:** A real number between 0 and 1; the desired specificity of the test.

**conf:** A real number between 0 and 1; the confidence level of the results. Can be obtained by subtracting the family-wise Type I error from 1.

**power:** A data frame containing the power computations. The rows are indexed by the sample size and the columns by the sensitivity.

**Methods**

**print(x, ...)** Print the power table x.

**summary(object, ...)** Summarize the power table object.

**Note**

See [biomarkerPowerTable](#) for examples.

**Author(s)**

Kevin R. Coombes <krc@silicovore.com>

**See Also**

[TailRankTest](#), [tailRankPower](#), [biomarkerPowerTable](#)

---

tailRankPower

*Power of the tail-rank test*

---

**Description**

Compute the significance level and the power of a tail-rank test.

**Usage**

```
tailRankPower(G, N1, N2, psi, phi, conf = 0.95,
              model=c("bb", "betabinom", "binomial"))
tailRankCutoff(G, N1, N2, psi, conf,
               model=c("bb", "betabinom", "binomial"),
               method=c('approx', 'exact'))
```

**Arguments**

G	An integer; the number of genes being assessed as potential biomarkers. Statistically, the number of hypotheses being tested.
N1	An integer; the number of "train" or "healthy" samples used.
N2	An integer; the number of "test" or "cancer" samples used.
psi	A real number between 0 and 1; the desired specificity of the test.
phi	A real number between 0 and 1; the sensitivity that one would like to be able to detect, conditional on the specificity.
conf	A real number between 0 and 1; the confidence level of the results. Can be obtained by subtracting the family-wise Type I error from 1.
model	A character string that determines whether significance and power are computed based on a binomial or a beta-binomial (bb) model.
method	A character string; either "exact" or "approx". The default is to use a Bonferroni approximation.

## Details

A power estimate for the tail-rank test can be obtained as follows. First, let  $X \sim \text{Binom}(N, p)$  denote a binomial random variable. Under the null hypothesis that cancer is not different from normal, we let  $p = 1 - \psi$  be the expected proportion of successes in a test of whether the value exceeds the  $\psi$ -th quantile. Now let

$$\alpha = P(X > x, |N, p)$$

be one such binomial measurement. When we make  $G$  independent binomial measurements, we take

$$\text{conf} = P(\text{all } G \text{ of the } X's \leq x | N, p).$$

(In our paper on the tail-rank statistic, we write everything in terms of  $\gamma = 1 - \text{conf}$ .) Then we have

$$\text{conf} = P(X \leq x | N, p)^G = (1 - \alpha)^G.$$

Using a Bonferroni-like approximation, we can take

$$\text{conf} = 1 - \alpha * G.$$

Solving for  $\alpha$ , we find that

$$\alpha = (1 - \text{conf})/G.$$

So, the cutoff that ensures that in multiple experiments, each looking at  $G$  genes in  $N$  samples, we have confidence level  $\text{conf}$  (or significance level  $\gamma = 1 - \text{conf}$ ) of no false positives is computed by the function `tailRankCutoff`.

The final point to note is that the quantiles are also defined in terms of  $q = 1 - \alpha$ , so there are lots of disfiguring "1's" in the implementation.

Now we set  $M$  to be the significance cutoff using the procedure detailed above. A gene with sensitivity  $\phi$  gets detected if the observed number of cases above the threshold is greater than or equal to  $M$ . The `tailRankPower` function implements formula (1.3) of our paper on the tail-rank test.

## Value

`tailRankCutoff` returns an integer that is the maximum expected value of the tail rank statistic under the null hypothesis.

`tailRankPower` returns a real number between 0 and 1 that is the power of the tail-rank test to detect a marker with true sensitivity equal to  $\phi$ .

## Author(s)

Kevin R. Coombes <krc@silicovore.com>

## See Also

[TailRankTest](#), [tailRankPower](#), [biomarkerPowerTable](#), [matrixMean](#), [toleranceBound](#)

**Examples**

```

psi.0 <- 0.99
confide <- rev(c(0.8, 0.95, 0.99))
nh <- 20
ng <- c(100, 1000, 10000, 100000)
ns <- c(10, 20, 50, 100, 250, 500)
formal.cut <- array(0, c(length(ns), length(ng), length(confide)))
for (i in 1:length(ng)) {
  for (j in 1:length(ns)) {
    formal.cut[j, i, ] <- tailRankCutoff(ng[i], nh, ns[j], psi.0, confide)
  }
}
dimnames(formal.cut) <- list(ns, ng, confide)
formal.cut

phi <- seq(0.1, 0.7, by=0.1)
N <- c(10, 20, 50, 100, 250, 500)
pows <- matrix(0, ncol=length(phi), nrow=length(N))
for (ph in 1:length(phi)) {
  pows[, ph] <- tailRankPower(10000, nh, N, 0.95, phi[ph], 0.9)
}
pows <- data.frame(pows)
dimnames(pows) <- list(as.character(N), as.character(round(100*phi)))
pows

```

---

TailRankTest

*The Tail-Rank Test*


---

**Description**

Perform a tail-rank test to find candidate biomarkers in a microarray data set.

**Usage**

```

TailRankTest(data, classes, specificity = 0.95, tolerance = 0.50,
             model=c("bb", "betabinomial", "binomial"),
             confidence = 0.95, direction = "up")

```

**Arguments**

<code>data</code>	A matrix or data.frame containing numerical measurements on which to perform the tail-rank test.
<code>classes</code>	A logical vector or factor splitting the data into two parts. The length of this vector should equal the number of columns in the data. The TRUE portion (or the first level of the factor) represents a "base" or "healthy" group of samples; the other samples are the "test" or "cancer" group.
<code>specificity</code>	a real number between 0 and 1; the desired specificity used in the test to estimate a quantile from the "base" group. This is an optional argument with default value 0.95.

tolerance	a real number between 0 and 1; the upper tolerance bound used to estimate the threshold. This is an optional argument with default value 0.90.
model	a character string that determines whether significance comes from a binomial model or a beta-binomial (bb) model.
confidence	a real number between 0 and 1; the confidence level that there are no false positives. This is an optional argument with default value 0.50, which is equivalent to ignoring the tolerance.
direction	a character string representing the direction of the test; can be "up", "down", or "two-sided". The default value is "up".

### Details

This function computes the tail rank statistic for each gene (viewed as one row of the data matrix). The data is split into two groups. The first ("base") group is used to estimate a tolerance bound (defaults to 50%) on a specific quantile (defaults to 95%) of the distribution of each gene. The tail-rank statistic is defined as the number of samples in the second ("test") group that lie outside the bound. The test can be applied in the "up", "down", or "two-sided" direction, depending on the kinds of markers being sought. Also computes the cutoff for significance based on a confidence level that is "1 - FWER" for a desired family-wise error rate.

### Value

The return value is an object of class [TailRankTest](#).

### Author(s)

Kevin R. Coombes <krc@silicovore.com>

### References

<http://bioinformatics.mdanderson.org>

### See Also

[TailRankTest-class](#), [tailRankPower](#), [biomarkerPowerTable](#), [toleranceBound](#)

### Examples

```
# generate some fake data to use in the example
nr <- 40000
nc <- 110
fake.data <- matrix(rnorm(nr*nc), ncol=nc)
fake.class <- rep(c(TRUE, FALSE), c(40, 70))

# perform the tail-rank test
null.tr <- TailRankTest(fake.data, fake.class)

# get a summary of the results
summary(null.tr)
```

```

# plot a histogram of the statistics
hist(null.tr, overlay=TRUE)

# get the actual statistics
stats <- getStatistic(null.tr)

# get a vector that selects the "positive" calls for the test
is.marker <- as.logical(null.tr)

# the following line should evaluate to the number of rows, nr = 40000
sum( is.marker == (stats > null.tr@cutoff) )

```

---

TailRankTest-class      *The TailRankTest Class*

---

## Description

This is the class representation for the results of a tail-rank test to find biomarkers in a microarray data set. It includes methods for summarizing and plotting the results of the test.

## Creating objects

Although objects can be created, as usual, using `new`, the only reliable way to create valid objects is to use the [TailRankTest](#) function. See the description of that function for details on how the tail-rank test works.

## Slots

**statistic:** a numeric vector containing the tail-rank statistic for each row (gene) in a microarray data set

**direction:** a character string representing the direction of the test; can be "up", "down", or "two-sided"

**N1:** an integer; the number of samples in the "base" or "healthy" group

**N2:** an integer; the number of samples in the "test" or "cancer" group

**specificity:** a real number between 0 and 1; the desired specificity used in the test to estimate a quantile from the "base" group

**tolerance:** a real number between 0 and 1; the upper tolerance bound used to estimate the threshold

**confidence:** a real number between 0 and 1; the confidence level that there are no false positives

**cutoff:** an integer; the maximum expected value of the statistic under the null hypothesis

**model:** a character string describing the model (binomial or beta-binomial) used to decide on cutoffs for significance

**tau:** a numeric vector or NULL; gene-by-gene upper bounds for significance

**rho:** a numeric vector or NULL; gene-by-gene lower bounds for significance

## Methods

**summary(object, ...)** Display a summary of the TailRankTest object

**hist(x, overlay, ...)** Plot a histogram of the statistic in the TailRankTest object x. The optional argument `overlay` is a logical flag. If `overlay=TRUE`, then the histogram is overlain with a curve representing the null distribution. The default value of `overlay` is `FALSE`.

**as.logical(x, ...)** Convert the TailRankTest object x into a logical vector, which takes on a `TRUE` value whenever the tail-rank statistic exceeds the significance cutoff.

**getStatistic(object, ...)** Obtain the vector of tail-rank statistics contained in object.

## Author(s)

Kevin R. Coombes <krc@silicovore.com>

## See Also

[TailRankTest](#), [tailRankPower](#), [biomarkerPowerTable](#), [matrixMean](#), [toleranceBound](#)

## Examples

```
# generate some fake data to use in the example
nr <- 40000
nc <- 110
fake.data <- matrix(rnorm(nr*nc), ncol=nc)
fake.class <- rep(c(TRUE, FALSE), c(40, 70))

# perform the tail-rank test
null.tr <- TailRankTest(fake.data, fake.class)

# get a summary of the results
summary(null.tr)

# plot a histogram of the statistics
hist(null.tr, overlay=TRUE)

# get the actual statistics
stats <- getStatistic(null.tr)

# get a vector that selects the "positive" calls for the test
is.marker <- as.logical(null.tr)

# the following line should evaluate to the number of rows, nr = 40000
sum( is.marker == (stats > null.tr@cutoff) )
```

---

**TailRankTest-methods** *Methods for TailRankTest objects*

---

**Description**

This file describes the methods for an object of the class [TailRankTest](#) class.

**Usage**

```
## S4 method for signature 'TailRankTest'  
summary(object, ...)  
## S4 method for signature 'TailRankTest'  
hist(x, overlay = FALSE,  
      xlab = "tail-rank statistic", main = "", ...)  
## S4 method for signature 'TailRankTest'  
as.logical(x, ...)  
## S4 method for signature 'TailRankTest'  
getStatistic(object,...)
```

**Arguments**

x	A <a href="#">TailRankTest</a> object
object	A <a href="#">TailRankTest</a> object
overlay	An optional logical flag; defaults to FALSE.
xlab	A character string
main	A character string
...	Extra graphical parameters

**Value**

as.logical	Returns a logical vector. TRUE values pick out candidate biomarkers where the tail-rank test statistic exceeds the significance cutoff.
getStatistic	Returns the vector of tail-rank statistics contained in object.
hist	Invisibly returns the <a href="#">TailRankTest</a> object.
summary	Invisibly returns the <a href="#">TailRankTest</a> object.

**Author(s)**

Kevin R. Coombes <krc@silicovore.com>

**See Also**

[TailRankTest-class](#), [TailRankTest](#), [tailRankPower](#)

**Examples**

```
# generate some fake data to use in the example
nr <- 40000
nc <- 110
fake.data <- matrix(rnorm(nr*nc), ncol=nc)
fake.class <- rep(c(TRUE, FALSE), c(40, 70))

# build an object
null.tr <- TailRankTest(fake.data, fake.class)

# summarize the object
summary(null.tr)

# plot a histogram
hist(null.tr)
hist(null.tr, breaks=70, col='blue', overlay=TRUE)

# get a logical vector that can select those markers
# identified by the test
selector <- as.logical(null.tr)
```

---

toleranceBound

*Upper tolerance bounds on normal quantiles*


---

**Description**

The function `toleranceBound` computes theoretical upper tolerance bounds on the quantiles of the standard normal distribution. These can be used to produce reliable data-driven estimates of the quantiles in any normal distribution.

**Usage**

```
toleranceBound(psi, gamma, N)
```

**Arguments**

<code>psi</code>	A real number between 0 and 1 giving the desired quantile
<code>gamma</code>	A real number between 0 and 1 giving the desired tolerance bound
<code>N</code>	An integer giving the number of observations used to estimate the quantile

**Details**

Suppose that we collect  $N$  observations from a normal distribution with unknown mean and variance, and wish to estimate the 95th percentile of the distribution. A simple point estimate is given by  $\tau = \bar{X} + 1.68s$ . However, only the mean of the distribution is less than this value 95% of the time. When  $N = 40$ , for example, almost half of the time (43.5%), fewer than 95% of the observed values will be less than  $\tau$ . This problem is addressed by constructing a statistical tolerance interval (more precisely, a one-sided tolerance bound) that contains a given fraction,  $\psi$ , of the population

with a given confidence level,  $\gamma$  [Hahn and Meeker, 1991]. With enough samples, one can obtain distribution-free tolerance bounds [op.\ cit., Chapter 5]. For instance, one can use bootstrap or jackknife methods to estimate these bounds empirically.

Here, however, we assume that the measurements are normally distributed. We let  $\bar{X}$  denote the sample mean and let  $s$  denote the sample standard deviation. The upper tolerance bound that,  $100\gamma\%$  of the time, exceeds  $100\psi\%$  of  $G$  values from a normal distribution is approximated by  $X_U = \bar{X} + k_{\gamma,\psi}s$ , where

$$k_{\gamma,\psi} = \frac{z_{\psi} + \sqrt{z_{\psi}^2 - ab}}{a},$$

$$a = 1 - \frac{z_{1-\gamma}^2}{2N - 2},$$

$$b = z_{\psi}^2 - \frac{z_{1-\gamma}^2}{N},$$

and, for any  $\pi$ ,  $z_{\pi}$  is the critical value of the normal distribution that is exceeded with probability  $\pi$  [Natrella, 1963].

### Value

Returns the value of  $k_{\gamma,\psi}$  with the property that the  $\psi$ th quantile will be less than the estimate  $X_U = \bar{X} + k_{\gamma,\psi}s$  (based on  $N$  data points) at least  $100\gamma\%$  of the time.

### Note

Lower tolerance bounds on quantiles with  $\psi$  less than one-half can be obtained as  $X_U = \bar{X} - k_{\gamma,1-\psi}s$ ,

### Author(s)

Kevin R. Coombes <krc@silicovore.com>

### References

- Natrella, M.G. (1963) *Experimental Statistics*. NBS Handbook 91, National Bureau of Standards, Washington DC.
- Hahn, G.J. and Meeker, W.Q. (1991) *Statistical Intervals: A Guide for Practitioners*. John Wiley and Sons, Inc., New York.

### Examples

```
N <- 50
x <- rnorm(N)
tolerance <- 0.90
quant <- 0.95
tolerance.factor <- toleranceBound(quant, tolerance, N)

# upper 90% tolerance bound for 95th percentile
tau <- mean(x) + sd(x)*tolerance.factor
```

```
# lower 90% tolerance bound for 5th percentile
rho <- mean(x) - sd(x)*tolerance.factor

# behavior of the tolerance bound as N increases
nn <- 10:100
plot(nn, toleranceBound(quant, tolerance, nn))

# behavior of the bound as the tolerance varies
xx <- seq(0.5, 0.99, by=0.01)
plot(xx, toleranceBound(quant, xx, N))
```

# Index

- \* **design**
    - BMPT-class, 5
  - \* **hstest**
    - BetaBinomial, 2
    - biomarkerPowerTable, 3
    - tailRankPower, 6
    - TailRankTest, 8
    - TailRankTest-class, 10
    - toleranceBound, 13
  - \* **manip**
    - TailRankTest-methods, 12
  - \* **multivariate**
    - biomarkerPowerTable, 3
    - tailRankPower, 6
    - TailRankTest, 8
    - TailRankTest-class, 10
  - \* **nonparametric**
    - biomarkerPowerTable, 3
    - tailRankPower, 6
    - TailRankTest, 8
    - TailRankTest-class, 10
  - \* **univar**
    - BetaBinomial, 2
    - biomarkerPowerTable, 3
    - tailRankPower, 6
    - TailRankTest, 8
    - TailRankTest-class, 10
    - toleranceBound, 13
- as.logical, TailRankTest-method  
(TailRankTest-methods), 12
- BetaBinomial, 2  
biomarkerPowerTable, 3, 4–7, 9, 11  
BMPT, 4  
BMPT (BMPT-class), 5  
BMPT-class, 5
- dbb (BetaBinomial), 2  
dbeta, 3
- dbinom, 3
- getStatistic (TailRankTest-class), 10  
getStatistic, TailRankTest-method  
(TailRankTest-methods), 12
- hist, TailRankTest-method  
(TailRankTest-methods), 12
- matrixMean, 4, 7, 11
- pbb (BetaBinomial), 2  
print, BMPT-method (BMPT-class), 5
- qbb (BetaBinomial), 2
- rbb (BetaBinomial), 2
- summary, BMPT-method (BMPT-class), 5  
summary, TailRankTest-method  
(TailRankTest-methods), 12
- tailRankCutoff (tailRankPower), 6  
tailRankPower, 4, 6, 6, 7, 9, 11, 12  
TailRankTest, 4, 6, 7, 8, 9–12  
TailRankTest-class, 10  
TailRankTest-methods, 12  
toleranceBound, 4, 7, 9, 11, 13