`biomarker.power.table` *Power tables for the tail-rank test*

Description

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

Usage

```
biomarker.power.table(G)
biomarker.power.table(G, N = seq(25, 250, by = 25))
biomarker.power.table(G, psi = c(0.95, 0.99))
biomarker.power.table(G, conf = 0.99)
biomarker.power.table(G, phi = seq(0.1, 0.5, by = 0.05))
biomarker.power.table(G, N, psi, conf, phi)
```

Arguments

`G` An integer; the number of genes being assessed as potential biomarkers. Statistically, the number of hypotheses being tested.

`N` 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.

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 <kcoombes@mdanderson.org>

See Also

`tail.rank.test`, `tail.rank.power`, `biomarker.power.table`, `matrix.mean`, `tol.bound`

Examples

```
stuff <- biomarker.power.table(10000,
                               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)
```

bmpt-class *The bmpt Class*

Description

A class for producing BioMarker Power Tables (bmpt), and methods for accessing them. This class is primarily an implementation detail for the function `biomarker.power.table`.

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 `biomarker.power.table`.

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 `biomarker.power.table` for examples.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

`tail.rank.test`, `tail.rank.power`, `biomarker.power.table`

`clinical.info` *Experimental info for the prostate cancer data set*

Description

This data set provides experimental and clinical information about the (partial) prostate cancer data set included for demonstration purposes as part of the tail.rank.test package. The experiments were two-color glass microarrays printed at Stanford.

Usage

```
data(clinical.info)
```

Format

A data frame with 112 observations on the following 6 variables.

- **Arrays** A factor containing the barcode of the microarray on which the experiment was performed. Each of the 112 entries should be distinct.
- **Reference** A factor describing the reference sample used in each experiment. This was a common reference, so the identifiers here are not meaningful.

Sample A factor identifying the test sample in each experiment. These match the codes published in the original paper.

Status A factor with three levels identifying normal prostate (N), prostate cancer (T), or lymph node metastasis (L).

Subgroups A factor with five levels: I II III N O. These correspond to the groups found in the original paper using clustering.

ChipType a factor with levels new or old. At least two different print designs of microarrays were used in this experiment; this factor identifies the design.

Source

The data was originally described in the paper by Lapointe et al., and downloaded from the Stanford Microarray Database http://genome-www5.stanford.edu/.

References

Lapointe J et al. (2004) Gene expression profiling identifies clinically relevant subtypes of prostate cancer. *Proc Natl Acad Sci U S A*, 101, 811–816.

See Also

expression.data, gene.info, tail.rank.test

Examples

```
data(clinical.info)
summary(clinical.info)
```

expression.data *Microarray expression data on prostate cancer*

Description

A subset of the microarray data from a study of prostate cancer at Stanford is supplied as demo data with the tail.rank.test package.

Usage

```
data(expression.data)
```

Format

A data frame with 2000 observations on the 112 variables. Each column represent a different patient sample, as described in the accompanying data.frame called clinical.info.

Details

This data set contains normalized microarray expression data on 2000 randomly selected genes from a prostate cancer data set. The study was originially described in a publication by Lapointe et al. The experiments were performed on two-color glass microarrays printed at Stanford and available from the Stanford Microarray Database. We donwnloaded the raw data and preprocessed it. In particular,after background correction and loess normalization, we computed log ratios between the channels. We then randomly selected 2000 of the 42129 spots to include as demonstration data here.

Source

http://genome-www5.stanford.edu/.

References

Lapointe J et al. (2004) Gene expression profiling identifies clinically relevant subtypes of prostate cancer. *Proc Natl Acad Sci U S A*, 101, 811–816.

See Also

clinical.info, gene.info, tail.rank.test

Examples

```
data(expression.data)
summary(expression.data)
```

---

gene.info *Gene information for the prostate cancer data set*

Description

This data set provides information about the genes included with the (partial) prostate cancer data set as part of the tail.rank.test package.

Usage

```
data(gene.info)
```

Format

A data frame with 2000 observations on the following 6 variables.

ArrayI.Spot a numeric vector; where is this clone spotted on the old arrays

ArrayII.Spot a numeric vector; where is this clone spotted on the new arrays

Clone.ID a factor; the IMAGE clone identifier

Gene.Symbol a factor; the official gene symbol

Cluster.ID a factor; the UniGene cluster number

Accession a factor; the GenBanlk accession number

Source

The data was originally described in the paper by Lapointe et al., and downloaded from the Stanford Microarray Database http://genome-www5.stanford.edu/. We randomly selected 2000 of the 42129 spots to include as demonstration data here.

References

Lapointe J et al. (2004) Gene expression profiling identifies clinically relevant subtypes of prostate cancer. *Proc Natl Acad Sci U S A*, 101, 811–816.

See Also

`clinical.info`, `expression.data`, `tail.rank.test`

Examples

```
data(gene.info)
summary(gene.info)
```

`matrix.mean` *Compute row-wise mean and variance using matrix operations*

Description

In large data sets, like those arising from microarray or proteomics experiments, it is often necessary to compute the mean and variance for each row among many thousands. The computations can be significantly faster if vectorized instead of using the inherent loop in an `apply` statement.

Usage

```
matrix.mean(x)
matrix.var(x, xmean)
```

Arguments

`x`	A matrix or data.frame containing numeric values.
`xmean`	A vector containing the means of the rows in `x`.

Value

A vector containing a number of entries equal to the number of rows of `x`, where each entry is either the mean or the variance of the corresponding row.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

apply

Examples

```
nr <- 10000
nc <- 40
fake.data <- matrix(rnorm(nr*nc), nrow=nr)
fake.class <- rep(c('H', 'C'), each=20)

H.n <- sum(fake.class=='H')
C.n <- sum(fake.class=='C')

H.mean <- matrix.mean(fake.data[, fake.class=='H'])
H.var <- matrix.var(fake.data[, fake.class=='H'], H.mean)

C.mean <- matrix.mean(fake.data[, fake.class=='C'])
C.var <- matrix.var(fake.data[, fake.class=='C'], C.mean)

pooled.sd <- sqrt( (H.var*(H.n - 1) + C.var*(C.n - 1))/(H.n + C.n - 2) )

t.statistics <- (C.mean - H.mean)/pooled.sd/sqrt(1/H.n + 1/C.n)
```

tail.rank.power *Power of the tail-rank test*

Description

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

Usage

```
tail.rank.power(G, N, psi, phi, conf = 0.95)
tail.rank.cutoff(G, N, psi, conf, method = "approx")
```

Arguments

G	An integer; the number of genes being assessed as potnetial biomarkers. Statistically, the number of hypotheses being tested.
N	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.
method	A character string; either ”exact” or ”approx”. The deafult is to use a Bonferroni approximation.

Details

A power estimate for the tail-rank test can be obtained as follows. First, let X Binom(N,p) denote a binomial random variable. Under the null hypotheis 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

$$conf = P(allGoftheX's \leq x|N, p).$$

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

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

Using a Bonferroni-like approximation, we can take

$$conf \ = 1 - \alpha * G.$$

Solving for α, we find that

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

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

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 ϕ gets detected if the observed number of cases above the threshold is greater than or equal to M. The `tail.rank.power` function implements formula (1.3) of our paper on the tail-rank test.

Value

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

`tail.rank.power` returns a real numbe 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 <kcoombes@mdanderson.org>

See Also

`tail.rank.test`, `tail.rank.power`, `biomarker.power.table`, `matrix.mean`, `tol.bound`

Examples

```
psi.0 <- 0.99
confide <- rev(c(0.8, 0.95, 0.99))
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, ] <- tail.rank.cutoff(ng[i], 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] <-  tail.rank.power(10000, N, 0.95, phi[ph], 0.9)
}
pows <- data.frame(pows)
dimnames(pows) <- list(as.character(N), as.character(round(100*phi)))
pows
```

---

tail.rank.test-class *The tail.rank.test 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 `tail.rank.test` function. See the description of that function for details on how the tail-rank test works.

Slots

- `statistic`: a numeric vector containng 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 numnber 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

Methods

summary(object, ...) Display a summary of the tail.rank.test `object`

hist(x, overlay, ...) Plot a histogram of the statistic in the tail.rank.test 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 tail.rank.statistic obejct `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 <kcoombes@mdanderson.org>

See Also

`tail.rank.test`, `tail.rank.power`, `biomarker.power.table`, `matrix.mean`, `tol.bound`

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 <- tail.rank.test(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) )
```

tail.rank.test-methods

Methods for tail.rank.test objects

Description

This file describes the methods for an object of the class `tail.rank.test` class.

Usage

```
## S4 method for signature 'tail.rank.test':
summary(object, ...)
## S4 method for signature 'tail.rank.test':
hist(x, overlay, ...)
## S4 method for signature 'tail.rank.test':
as.logical(x, ...)
## S4 method for signature 'tail.rank.test':
getStatistic(object,...)
```

Arguments

`x`	A `tail.rank.test` `object`
`object`	A `tail.rank.test` `object`
`overlay`	An optional logical flag; defaults to `FALSE`.
`...`	Extra graphical parameters

Value

`this-is-escaped-codenormal-bracket34bracket-normal`
Returns a logical vector. `TRUE` values pick out candidate biomarkers where the tail-rank test statistic exceeds the significance cutoff.

`this-is-escaped-codenormal-bracket38bracket-normal`
Returns the vector of tail-rank statistics contained in `object`.

`this-is-escaped-codenormal-bracket42bracket-normal`
Invisibly returns the tail.rank.test object.

`this-is-escaped-codenormal-bracket45bracket-normal`
Invisibly returns the tail.rank.test object.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

`tail.rank.test-class`, `tail.rank.test`, `tail.rank.power`

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 <-  tail.rank.test(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)
```

`tail.rank.test` *The Tail-Rank Test*

Description

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

Usage

```
tail.rank.test(data, split)
tail.rank.test(data, split, direction = "down")
tail.rank.test(data, split, specificity = 0.95, tolerance = 0.9, confidence = 0.95, direction =
```

Arguments

`data` A matrix or data.frame containing numerical measurements on which to perform the tail-rank test.

`split` 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.

`specificity` 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.

`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 0.95.

`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 90%) on a specific quantile (defaults to 95%) of the distribution of each gene. The tail-rank statistic is the 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 tail.rank.test.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

References

http://bioinformatics.mdanderson.org

See Also

`tail.rank.test-class`, `tail.rank.power`, `biomarker.power.table`, `tol.bound`

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 <- tail.rank.test(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) )
```

tol.bound *Upper tolerance bounds on normal quantiles*

Description

The function `tol.bound` 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

```
tol.bound(psi, gamma, N)
```

Arguments

`psi`	A real number between 0 and 1 giving the desired quantile
`gamma`	A real number between 0 and 1 giving the desired tolerance bound
`N`	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 τ. This problem is addressed by constructing a statistical tolerance interval (more precisely, a one-sided tolerance bound) that contains a given fraction, ψ, of the population with a given confidence level, γ [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 π, z_π is the critical value of the normal distribution that is exceeded with probability π [Natrella, 1963].

Value

Returns the value of $k_{\gamma,\psi}$ with the property that the ψth 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 <kcoombes@mdanderson.org>

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 <- tol.bound(quant, tolerance, N)

# upper 90
tau <- mean(x) + sd(x)*tolerance.factor

# lower 90
rho <- mean(x) - sd(x)*tolerance.factor

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

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