Package ‘LPE’
March 22, 2017

Version 1.48.0

Title Methods for analyzing microarray data using Local Pooled Error (LPE) method

Author Nitin Jain <emailnitinjain@gmail.com>, Michael O’Connell <michaelo@warath.com>, Jae K. Lee <jaeklee@virginia.edu>. Includes R source code contributed by HyungJun Cho <hcho@virginia.edu>

Maintainer Nitin Jain <emailnitinjain@gmail.com>

Description This LPE library is used to do significance analysis of microarray data with small number of replicates. It uses resampling based FDR adjustment, and gives less conservative results than traditional 'BH' or 'BY' procedures. Data accepted is raw data in txt format from MAS4, MAS5 or dChip. Data can also be supplied after normalization. LPE library is primarily used for analyzing data between two conditions. To use it for paired data, see LPEP library. For using LPE in multiple conditions, use HEM library.

Imports stats

License LGPL


Depends R (>= 2.10)

biocViews Microarray, DifferentialExpression

NeedsCompilation no

R topics documented:

am.trans .............................................................. 2
baseOlig.error ..................................................... 3
baseOlig.error.step1 ............................................... 5
baseOlig.error.step2 ............................................... 6
fdr.adjust ........................................................... 8
fixbounds.predict.smooth.spline .................................. 9
iqr ................................................................. 10
Transform replicated arrays into (A,M) format

Description

Transforms expression intensity measurements for replicated arrays of a single experimental condition into (A,M) format: $A = (x_i + x_j)/2$, $M = (x_i - x_j)$ where $x_1, x_2, ..., x_n$ are individual chips. This function is used in the estimation of within-bin variances in the LPE function, and not typically by the user.

Usage

am.trans(y)

Arguments

y

y is an ngene by n matrix of expression intensity measurements for replicated arrays under a single experimental condition.

Value

Returns matrix with 2 columns cbind(A,M) and rows comprising all permutations of individual chip columns of the input matrix y. Note that for each pair of chips M is calculated twice, once for $x_i - x_j$ and once for $x_j - x_i$. The resulting matrix thus has twice the number of rows as the input matrix y.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays*, Bioinformatics, 1945-1951.

See Also

lpe

Examples

library(LPE)
# Loading the LPE library

# Test data with duplicates
temp1 <- matrix(c(1,20,1.5,23),nrow=2)
am.trans(temp1)
# It gives a matrix of (4*2) as only two permutations
# are possible for each row with duplicates (X1-X2, and X2-X1)

# Another test data with three replicates
temp2 <- matrix(c(1,20,1.5,23,0.8,19),nrow=2)
am.trans(temp2)
# Now it returns matrix of (12*2) as there are
# 6 possible permutations for each row with triplicates
# (X1-X2, X1-X3, X2-X3, X2-X1, X3-X1 and X3-X2)

baseOlig.error

Evaluates LPE variance function of M for quantiles of A within and experimental condition and then interpolates it for all genes.

Description

Calls baseOlig.error.step1 and baseOlig.error.step2 functions in order to calculate the baseline distribution.

Usage

baseOlig.error(y, stats=median, q=0.01, min.genes.int=10, div.factor=1)

Arguments

y y is a preprocessed matrix or data frame of expression intensities in which columns are expression intensities for a particular experimental condition and rows are genes.

stats It determines whether mean or median is to be used for the replicates

q q is the quantile width; q=0.01 corresponds to 100 quantiles i.e. percentiles. Bins/quantiles have equal number of genes and are split according to the average intensity A.

min.genes.int Determines the minimum number of genes in a subinterval for selecting the adaptive intervals.

div.factor Determines the factor by which sigma needs to be divided for selecting adaptive intervals.
baseOlig.error

**Value**

Returns object of class baseOlig comprising a data frame with 2 columns: A and var M, and rows for each quantile specified. The A column contains the median values of A for each quantile/bin and the M columns contains the pooled variance of the replicate chips for genes within each quantile/bin.

**Author(s)**

Nitin Jain<nitin.jain@pfizer.com>

**References**


Jain et. al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays*, Bioinformatics, 1945-1951.


**See Also**

lpe

**Examples**

```r
# Loading the library and the data
library(LPE)
data(Ley)

dim(Ley)
# Gives 12488 by 7
Ley[1:3,]  # Returns
# ID c1 c2 c3 t1 t2 t3
# 1 AFFX-MurIL2_at 4.06 3.82 4.28 11.47 11.54 11.34
# 2 AFFX-MurIL10_at 4.56 2.79 4.83 4.25 3.72 2.94
# 3 AFFX-MurIL4_at 5.14 4.10 4.59 4.67 4.71 4.67
Ley[,2:7] <- preprocess(Ley[,2:7], data.type="MAS5")

subset <- 1:1000
Ley.subset <- Ley[subset,]

# Finding the baseline distribution of subset of the data
# condition one (3 replicates)
var.1 <- baseOlig.error(Ley.subset[,2:4], q=0.01)
dim(var.1)
# Returns a matrix of 1000 by 2 (A,M) format, equal to the nrow(data)
```
Description

Genes are placed in bins/quantiles according to their average expression intensity. The function baseOlig.error calculates a pooled variance of M for genes within these bins/quantiles of A for the replicates of the experimental condition contained in y. Here the assumption is that variance of the genes in each interval is similar.

Usage

baseOlig.error.step1(y, stats=median, q=0.01, df=10)

Arguments

y  y is a preprocessed matrix or data frame of expression intensities in which columns are expression intensities for a particular experimental condition and rows are genes.

stats  It determines whether mean or median is to be used for the replicates

q  q is the quantile width; q=0.01 corresponds to 100 quantiles i.e. percentiles. Bins/quantiles have equal number of genes and are split according to the average intensity A.

df  df stands for degrees of freedom. It is used in smooth.spline function to interpolate the variances of all genes. Default value is 10.

Value

Returns object of class baseOlig, comprising a data frame with 2 columns: A and var M. The A column contains the median values of each gene and the M columns contains the corresponding variance. Number of rows of the data-frame is same as that of the number of genes.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.


See Also

lpe
Examples

```r
# Loading the library and the data
library(LPE)
data(Ley)

dim(Ley)
# Gives 12488 by 7
Ley[1:3,]
# Returns
# ID c1 c2 c3 t1 t2 t3
# 1 AFFX-MurIL2_at 4.06 3.82 4.28 11.47 11.54 11.34
# 2 AFFX-MurIL10_at 4.56 2.79 4.83 4.25 3.72 2.94
# 3 AFFX-MurIL4_at 5.14 4.10 4.59 4.67 4.71 4.67

Ley[1:1000,2:7] <- preprocess(Ley[1:1000,2:7], data.type="MAS5")
# Finding the baseline distribution of subset of the data
# condition one (3 replicates)
var.1 <- baseOlig.error.step1(Ley[1:1000,2:4], q=0.01)
dim(var.1)
# Returns a matrix of 1000 by 2 (A,M) format
```

---

**baseOlig.error.step2**

Evaluates LPE variance function of M for quantiles of A within and experimental condition. It is based on the adaptive number of intervals.

**Description**

Similar to baseOlig.error.step1 function, except that now the number of bins are chosen adaptively instead of fixed 100.

**Usage**

```r
baseOlig.error.step2(y, baseOlig.error.step1.res, df=10, stats=median, min.genes.int=10, div.factor=1)
```

**Arguments**

- `y`: y is a preprocessed matrix or data frame of expression intensities in which columns are expression intensities for a particular experimental condition and rows are genes.
- `baseOlig.error.step1.res`: It is the result obtained from baseOlig.error.step1 function, in which number of bins are fixed=100
- `df`: df stands for degrees of freedom. It is used in smooth.spline function to interpolate the variances of all genes. Default value is 10.
- `stats`: It determines whether mean or median is to be used for the replicates
- `min.genes.int`: Determines the minimum number of genes in a subinterval for selecting the adaptive intervals.
- `div.factor`: Determines the factor by which sigma needs to be divided for selecting adaptive intervals.
Value

Returns object of class baseOlig comprising a data frame with 2 columns: A and var M, and rows for each quantile specified. The A column contains the median values of A for each quantile/bin and the M columns contains the pooled variance of the replicate chips for genes within each quantile/bin.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.


See Also

lpe

Examples

# Loading the library and the data
library(LPE)
data(Ley)

dim(Ley)
# Gives 12488 by 7
Ley[1:3,]
# Returns
# ID c1 c2 c3 t1 t2 t3
# 1 AFFX-MurIL2_at 4.06 3.82 4.28 11.47 11.54 11.34
# 2 AFFX-MurIL10_at 4.56 2.79 4.83 4.25 3.72 2.94
# 3 AFFX-MurIL4_at 5.14 4.10 4.59 4.67 4.71 4.67

Ley[1:1000,2:7] <- preprocess(Ley[1:1000,2:7],data.type="MAS5")
# Finding the baseline distribution of subset of the data
# condition one (3 replicates)
var.1 <- baseOlig.error.step1(Ley[1:1000,2:4], q=0.01, df=10)
dim(var.1)
var.11 <- baseOlig.error.step2(Ley[1:1000,2:4], var.1, df=10)
# Returns a matrix of 1000 by 2 (A,M) format
fdr.adjust  FDR adjustment procedures

Description

Based on the type of adjustment, eg: resampling, BH, BY, etc, calls appropriate functions for fdr adjustment.

Usage

fdr.adjust(lpe.result, adjp="resamp", target.fdr=c(10^-3, seq(0.01, 0.10, 0.01), 0.15, 0.20, 0.50), iterations=5, ALL=FALSE)

Arguments

lpe.result  Data frame obtained from calling lpe function
adjp  Type of adjustment procedure. Can be "resamp", "BH", "BY", "Bonferroni" or "mix.all"
target.fdr  Desired FDR level (used only for resampling based adjustment)
iterations  Number of iterations for stable z-critical.
ALL  If TRUE, the FDR corresponding to all the z-statistics, i.e. for every gene intensity is given.

Details

Returns the output similar to lpe function, including adjusted FDR. BH and BY give Benjamini-Hochberg and Benjamini-Yekutieli adjusted FDRs (adopted from multtest procedure), Bonferroni adjusted p-values and "mix.all" gives SAM-like FDR adjustment. For further details on the comparisons of each of these methods, please see the reference paper (Rank-invariant resampling...) mentioned below. Users are encouraged to use FDR instead of Bonferroni adjusted p-value as initial cutoffs while selecting the significant genes. Bonferroni adjusted p-values are provided under Bonferroni method here just for the sake of completion for the users who want it.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays*, Bioinformatics, 1945-1951.

Examples

```r
# Loading the library and the data
library(LPE)
data(Ley)

dim(Ley)
# Gives 12488*7
# First column is ID.
Ley[,2:7] <- preprocess(Ley[,2:7], data.type="MAS5")

# Subsetting the data
subset.Ley <- Ley[1:1000,]

# Finding the baseline distribution of condition 1 and 2.
var.1 <- baseOlig.error(subset.Ley[,2:4], q=0.01)
var.2 <- baseOlig.error(subset.Ley[,5:7], q=0.01)

# Applying LPE
lpe.result <- lpe(subset.Ley[,2:4], subset.Ley[,5:7], var.1, var.2,
 probe.set.name=subset.Ley[,1])

final.result <- fdr.adjust(lpe.result, adjp="resamp", target.fdr=c(0.01, 0.05), iterations=1)
final.result
```

Fixbounds.predict.smooth.spline

Makes the predicted variance non negative

Description

Makes the predicted variance non negative

Usage

`fixbounds.predict.smooth.spline(object, x, deriv=0)`

Arguments

- `object`: variance from baseOlig.error function
- `x`: vector for which variance needs to be predicted
- `deriv`: derivative of the vector required, default =0

Value

Returns the predicted variance for the given vector based on the baseline error distribution. Maximum and minimum predicted values for the vector are same as those of baseline error distribution
Author(s)
Nitin Jain<nitin.jain@pfizer.com>

References
Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.

Examples
# Loading the library and the data
library(LPE)
data(Ley)
dim(Ley)
# Gives 12488*7
# First column is ID.

# Subsetting the data
subset.Ley <- Ley[1:1000,]
subset.Ley[,2:7] <- preprocess(subset.Ley[,2:7],data.type="MAS5")
# preprocess the data

# Finding the baseline distribution of condition 1 and 2.
var.1 <- baseOlig.error(subset.Ley[,2:4], q=0.01)
median.x <- apply(subset.Ley[,2:4], 1, median)
sf.x <- smooth.spline(var.1[, 1], var.1[, 2], df = 10)
var.test <- fixbounds.predict.smooth.spline(sf.x, median.x)$y

---

**iqr**

*Inter-quartile range*

**Description**

Finds inter-quartile range of the data = \{75th percentile - 25th percentile\}.

**Usage**

`iqr(x)`

**Arguments**

`x`  
`x` is a vector for which IQR has to be found.
Ley

Value

Returns a numeric value representing the difference of 75th percentile and 25th percentile of the vector. It is used for normalization across the chips - basic assumption is that net differential expression of the middle half of the genes in microarray experiment is zero, which is conservative assumption as typically only 5-10 differential expression.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.


See Also

lpe

Examples

library(LPE)

# Loading the LPE library

iqr(1:5)

Ley

Gene Expression Data from Mouse Immune response study, (2002)

Description

Affymetrix GeneChip (12488 genes and 3 different conditions, each with 3 replicates) experiment was conducted by Dr. Ley (2002) to understand mouse immune response study. For demonstration purposes, we show data from 2 conditions, i.e. 6 chips, only.

Usage

data(Ley)

Format

Matrix of 12488 genes by 7 columns. First column is the GeneID, next three columns are replicates in condition one, and last three columns are replicates in condition 2.

For details, contact. Dr. Klaus Ley, <kfl3f@virginia.edu>, http://hsc.virginia.edu/medicine/basic-sci/biomed/ley/
Author(s)
Nitin Jain <nitin.jain@pfizer.com>

References


---

lowess.normalize

Lowess normalization of the data (based on $M$ vs $A$ graph)

Description
All the chips are normalized w.r.t. 1st chip

Usage
lowess.normalize(x,y)

Arguments
x x is the chip data w.r.t. which other chips would be normalized
y y is the chip data which would be normalized

Value
Returns the lowess normalized chip intensity.

Author(s)
Nitin Jain <nitin.jain@pfizer.com>

References

Jain et. al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays*, Bioinformatics, 1945-1951.


See Also
lpe
Examples

```r
library(LPE)
# Loading the LPE library
data(Ley)
# Loading the data set
dim(Ley) #gives 12488 * 7
Ley[1:3,]

Ley[1:1000,2:7] <- preprocess(Ley[1:1000,2:7], data.type="MAS5")
Ley[1:3,]
```

**lpe**

*Evaluates local pooled error significance test*

**Description**

The local pooled error test attempts to reduce dependence on the within-gene estimates in tests for differential expression, by pooling error estimates within regions of similar intensity. Note that with the large number of genes there will be genes with low within-gene error estimates by chance, so that some signal-to-noise ratios will be large regardless of mean expression intensities and fold-change. The local pooled error attempts to avert this by combining within-gene error estimates with those of genes with similar expression intensity.

**Usage**

```r
lpe(x, y, basevar.x, basevar.y, df=10, array.type="olig",
    probe.set.name=NULL, trim.percent=5)
```

**Arguments**

- `x`: Replicated data from first experimental condition (as matrix or data-frame).
- `y`: Replicated data from second experimental condition (as matrix or data-frame).
- `basevar.x`: Baseline distribution of first condition obtained from function baseOlig.error
- `basevar.y`: Baseline distribution of second condition obtained from function baseOlig.error
- `df`: Degrees of freedom used in fitting smooth.spline to estimates of var.M for bins in A
- `array.type`: Currently supports oligo arrays
- `probe.set.name`: Gene IDs. By default if they are not provided then 1,2,3,... is assigned as GeneID
- `trim.percent`: Percent of (A, var.M) estimates to trim from low end of A

**Details**

The LPE test statistic numerator is the difference in medians between the two experimental conditions. The test statistic denominator is the combined pooled standard error for the two experimental conditions obtained by looking up the var.M from each baseOlig.error variance function. The conversion to p-values is based on the Gaussian distribution for difference if order statistics (medians). The user may select both the smoother degrees of freedom (smaller is smoother) and the trim percent to obtain a variance function to suit particular issues i.e. variability of genes with low expression intensity.
Value

Data frame including x, median of x, y, median of y, median difference of (x,y), pooled standard deviation of difference, LPE p-value, outlier flag, probability of an outlier within x or y.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.


Examples

```r
# Loading the library and the data
library(LPE)
data(Ley)
dim(Ley)
# Gives 12488x7
# First column is ID.

# Subsetting the data
subset.Ley <- Ley[1:1000,]
subset.Ley[,2:7] <- preprocess(subset.Ley[,2:7],data.type="MAS5")

# Finding the baseline distribution of condition 1 and 2.
var.11 <- baseOlig.error.step1(subset.Ley[,2:4])
var.1 <- baseOlig.error(subset.Ley[,2:4], q=0.01)
var.2 <- baseOlig.error(subset.Ley[,5:7], q=0.01)

# Applying LPE
lpe.result <- lpe(subset.Ley[,2:4],subset.Ley[,5:7], var.1, var.2, probe.set.name=subset.Ley[,1])
```

mt.rawp2adjp.LPE

Adjusted p-values for simple multiple testing procedures
**Description**

This function computes adjusted p-values for simple multiple testing procedures from a vector of raw (unadjusted) p-values. The procedures include the Bonferroni, Holm (1979), Hochberg (1988), and Sidak procedures for strong control of the family-wise Type I error rate (FWER), and the Benjamini & Hochberg (1995) and Benjamini & Yekutieli (2001) procedures for (strong) control of the false discovery rate (FDR).

**Usage**

```r
mt.rawp2adjp.LPE(rawp, proc=c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH", "BY"))
```

**Arguments**

- `rawp`: A vector of raw (unadjusted) p-values for each hypothesis under consideration. These could be nominal p-values, for example, from t-tables, or permutation p-values as given in `mt.maxT` and `mt.minP`. If the `mt.maxT` or `mt.minP` functions are used, raw p-values should be given in the original data order, `rawp[order(index)]`.

- `proc`: A vector of character strings containing the names of the multiple testing procedures for which adjusted p-values are to be computed. This vector should include any of the following: "Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH", "BY".

**Details**

Adjusted p-values are computed for simple FWER and FDR controlling procedures based on a vector of raw (unadjusted) p-values.

- **Bonferroni**: Bonferroni single-step adjusted p-values for strong control of the FWER.
- **Holm**: Holm (1979) step-down adjusted p-values for strong control of the FWER.
- **Hochberg**: Hochberg (1988) step-up adjusted p-values for strong control of the FWER (for raw (unadjusted) p-values satisfying the Simes inequality).
- **SidakSS**: Sidak single-step adjusted p-values for strong control of the FWER (for positive orthant dependent test statistics).
- **SidakSD**: Sidak step-down adjusted p-values for strong control of the FWER (for positive orthant dependent test statistics).
- **BY**: adjusted p-values for the Benjamini & Yekutieli (2001) step-up FDR controlling procedure (general dependency structures).

**Value**

A list with components

- `adjp`: A matrix of adjusted p-values, with rows corresponding to hypotheses and columns to multiple testing procedures. Hypotheses are sorted in increasing order of their raw (unadjusted) p-values.

- `index`: A vector of row indices, between 1 and `length(rawp)`, where rows are sorted according to their raw (unadjusted) p-values. To obtain the adjusted p-values in the original data order, use `adjp[order(index),]`. 
Author(s)

Sandrine Dudoit, http://www.stat.berkeley.edu/~sandrine,
Yongchao Ge, <gyc@stat.berkeley.edu>.

References


S. Dudoit, J. P. Shaffer, and J. C. Boldrick (Submitted). Multiple hypothesis testing in microarray experiments.


See Also

lpe

Examples

# Loading the library and the data
library(LPE)
data(Ley)
dim(Ley)
# Gives 12488x7
# First column is ID.

# Subsetting the data
subset.Ley <- Ley[1:1000,]
subset.Ley[,2:7] <- preprocess(subset.Ley[,2:7], data.type="MAS5")

# Finding the baseline distribution of condition 1 and 2.
var.1 <- baseOlig.error(subset.Ley[,2:4], q=0.01)
var.2 <- baseOlig.error(subset.Ley[,5:7], q=0.01)

# Applying LPE
lpe.result <- lpe(subset.Ley[,2:4], subset.Ley[,5:7], var.1, var.2, probe.set.name=subset.Ley[,1])
fdr.BH <- fdr.adjust(lpe.result, adjp="BH")
**n.genes.adaptive.int**

Calculates the number of genes in various intervals adaptively.

**Description**

Instead of dividing the genes equally in 100 intervals, this function divides them adaptively based on three rules: a) min. number of genes (default =10), b) max. number of genes = total/100; c) based on Median + fraction(SD) from the starting gene of each interval

**Usage**

```r
n.genes.adaptive.int(baseOlig.error.step1.res,
                      min.genes.int=10, div.factor=1)
```

**Arguments**

- `baseOlig.error.step1.res`: It is the result from baseOlig.error.step1 function.
- `min.genes.int`: It is the minimum number of genes in the interval, default=10.
- `div.factor`: (1/div.factor) is the fraction of Standard Deviation which we wish to include in each interval to calculate number of genes in each interval

**Value**

Returns a vector representing the number of genes in each interval.

**Author(s)**

Nitin Jain<nitin.jain@pfizer.com>

**References**


Jain et. al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays*, Bioinformatics, 1945-1951.


**See Also**

lpe
Examples

```r
# Loading the library and the data
library(LPE)
data(Ley)
dim(Ley)
# Gives 12488 by 7
Ley[1:3,]
# Returns
# ID            c1  c2  c3  t1  t2  t3
# 1 AFFX-MurIL2_at 4.06 3.82 4.28 11.47 11.54 11.34
# 2 AFFX-MurIL10_at 4.56 2.79 4.83 4.25 3.72 2.94
# 3 AFFX-MurIL4_at 5.14 4.10 4.59 4.67 4.67 4.67

Ley[1:1000,2:7] <- preprocess(Ley[1:1000,2:7],data.type="MAS5")
# Finding the baseline distribution of subset of the data
# condition one (3 replicates)
var.1 <- baseOlig.error.step1(Ley[1:1000,2:4], q=0.01)
dim(var.1)
# Returns a matrix of 1000 by 2 (A,M) format
n.genes.subint <- n.genes.adaptive.int(var.1, min.genes.int=10, div.factor=1)
```

permute

*Calculating all possible permutations of a vector*

Description

Given a vector, all possible combinations of vector are obtained

Usage

```r
permute(a)
```

Arguments

```r
a
```
a is any numeric vector.

Details

Used in am.trans. Does all permutations for columns within an experimental condition so that A and M can be calculated for all permutations of chips within a treatment.

Value

A vector containing the possible combinations.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>
References


Jain et al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays,* Bioinformatics, 1945-1951.


See Also

lpe

Examples

```r
# Loading LPE library
library(LPE)

# A test vector
permute(1:3)

# Returns a 2 by 3 matrix
# [,1] [,2] [,3]
# [1,] 2 3 1
# [2,] 3 1 2
```

```r
preprocess(x, data.type="MAS5", threshold=1, LOWESS=FALSE)
```

Arguments

- `x` x is the data-set which needs preprocessing.
- `data.type` Three types of data accepted in the current version: MAS4 (Microarray suite software), MAS5 and dChip.
- `threshold` threshold is the ‘thresholding value’ below which all data would be thresholded (default = 1).
- `LOWESS` LOWESS is a logical variable which determines if lowess normalization needs to be performed.
Value

Returns a data-set of same dimensions as that of the input data. It has IQR normalization for MAS4 and MAS5 data. Low intensities of MAS4, MAS5 and dChip data are thresholded to 1. Then data is transformed to base 2. If LOWESS normalization parameter is set as TRUE, then lowess normalization is performed.

Author(s)

Nitin Jain <nitin.jain@pfizer.com>

References


Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.


See Also

lpe

Examples

library(LPE)
# Loading the LPE library

data(Ley)
# Loading the data set
dim(Ley) #gives 12488 * 7
Ley[1:3,]

Ley[1:1000,2:7] <- preprocess(Ley[1:1000,2:7], data.type="MAS5", threshold=1, LOWESS=TRUE)
Ley[1:3,]

---

**Finding quartile range**

**Description**

Finds quartile range of the data (default is IQR = 75th percentile - 25th percentile).

**Usage**

```
quan.norm(x, percent=50)
```
quartile.normalize

Arguments

x  
   x is a vector for which quartile range has to be found.

percent  
   Percentage for which quartile range is needed

Value

Returns a numeric value representing the difference of 75th percentile and 25th percentile of the vector. It is used for normalization across the chips - basic assumption is that net differential expression of the middle half of the genes in microarray experiment is zero, which is conservative assumption as typically only 5-10 differential expression.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays, Bioinformatics, 1945-1951.


See Also

lpe

Examples

library(LPE)

# Loading the LPE library

quan.norm(1:5)
Value

Returns the normalized data based on quartile normalization

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Jain et. al. (2003) *Local pooled error test for identifying differentially expressed genes with a small number of replicated microarrays,* Bioinformatics, 1945-1951.


See Also

lpe

Examples

library(LPE)
# Loading the LPE library
data(Ley)

dim(Ley)
# Gives 12488*7
# First column is ID.

subset <- 1:1000
Ley[subset,2:7] <- quartile.normalize(Ley[subset,2:7], percent=50)

---

resamp.adj

**Resampling based fdr adjustment**

Description

Adjusts the fdr based on rank invariant genes

Usage

resamp.adj(x, y, q=0.01, iterations=5, min.genes.int=10)
Arguments

- **x**: Replicated data from first experimental condition (as matrix or data-frame).
- **y**: Replicated data from second experimental condition (as matrix or data-frame).
- **q**: The quantile width; q=0.01 corresponds to 100 quantiles.
- **iterations**: Number of iterations to be performed to obtain critical z-statistics.
- **min.genes.int**: Determines the minimum number of genes in a subinterval for selecting the adaptive intervals.

Details

Returns the z-statistics for the null distribution, obtained from resampling the rank invariant genes within each quantile. These z-statistic values are compared with z-statistics from the original data, and fdr is calculated.

Author(s)

Nitin Jain<nitin.jain@pfizer.com>

References


Examples

```r
# Loading the library and the data
library(LPE)
data(Ley)

dim(Ley)
# Gives 12488*7
# First column is ID.

# Subsetting the data
subset.Ley <- Ley[1:1000,]
subset.Ley[,2:7] <- preprocess(subset.Ley[,2:7], data.type="MAS5")

# Finding the baseline distribution of condition 1 and 2.
var.1 <- baseOlig.error(subset.Ley[,2:4], q=0.01)
var.2 <- baseOlig.error(subset.Ley[,5:7], q=0.01)

# Applying LPE
lpe.result <- lpe(subset.Ley[,2:4],subset.Ley[,5:7], var.1, var.2, probe.set.name=subset.Ley[,1])
```
z.stats.null <- resamp.adj(subset.Ley[,2:4], subset.Ley[,5:7], q=0.01, iterations=2, min.genes.int=10)
Index

*Topic datasets
  Ley, 11

*Topic htest
  mt.rawp2adjp.LPE, 14

*Topic methods
  am.trans, 2
  baseOlig.error, 3
  baseOlig.error.step1, 5
  baseOlig.error.step2, 6
  fdr.adjust, 8
  fixbounds.predict.smooth.spline, 9
  iqr, 10
  lowess.normalize, 12
  lpe, 13
  n.genes.adaptive.int, 17
  permute, 18
  preprocess, 19
  quan.norm, 20
  quartile.normalize, 21
  resamp.adj, 22

am.trans, 2
baseOlig.error, 3
baseOlig.error.step1, 5
baseOlig.error.step2, 6
fdr.adjust, 8
fixbounds.predict.smooth.spline, 9
iqr, 10
Ley, 11
lowess.normalize, 12
lpe, 3–5, 7, 11, 12, 13, 16, 17, 19–22
mt.rawp2adjp.LPE, 14
n.genes.adaptive.int, 17
permute, 18
preprocess, 19
quan.norm, 20
quartile.normalize, 21
resamp.adj, 22