Package ‘deGPS’

August 21, 2014

Type Package

Title Differential Expression Tests Based on Generalized Poisson Statistic

Version 1.0

Date 2014-02-14

Author Chen Chu

Maintainer Chen Chu <chuchen.blueblues@gmail.com>

Depends R (>= 3.0.0), foreach, doParallel

Suggests LPE, limma, edgeR

Description Use methods based on Generalized Poisson Distribution to do RNA-seq differential expression tests.

License GPL-2

Encoding latin1

R topics documented:

deGPS-package *Normalization and Two-group Differential Expression Test for RNA-seq Data*

Description

This package is proposed to analyze RNA-seq data in two steps: normalization and differential expression test.

New normalization methods based on generalized Poisson distribution are contained in the main analysis functions, in which `GP-Theta` is suggested. Other popular normalization methods, such as TMM, LOWESS, Quantile, is also availbale in the package. More than one method can be specified in one run, in which case the resulting p values are a p value matrix, with each column representing one method.

Differential expression test is designed for RNA-seq data, especially those of small sample size, based on permutation strategy. Note that deGPS can only handle DE tests between two groups, but the novel GP-based normalization methods can be applied on any RNA-seq data. More Details about the GP-based normalization method and the advantages and limitations of our DE test can be found in our article.

The permutation step may become time-consuming in large sample size context or in mRNA read count data analysis. Parallel computation is introduced in the main functions to deal with the computational burden. Note that in situations where parallel computation is not necessary, to force parallelling may result in more run time.

The package also contains function to generate GP distrbuted data to be an example of RNA-seq data. It has to be pointed out that the simulated data in this package is FAR AWAY from real RNA-seq data, it is just simple GP distributed samples. For more appropriately simulated RNA-seq, `compcodeR` package is suggested, or, alternatively, you can generate H0 and H1 data from real data. A simple example is given in the following example session. More R codes referring to the real data based simulation or compcodeR based simulation can be found in the supplementary materials of our article.

Please do not hesitate to contact the author if you have any questions or find any flaws in the package.

Details

Package:	deGPS
Type:	Package
Version:	1.0
Date:	2014-02-24
License:	GPL-2

To do DE test for miRNA, call GPSmle. To do DE test for mRNA, call deGPS_mRNA. If only the normalization step is required, call GPSmle and specify `type` as "normalization".

Some denotations used in this manual are as follow:

- `nSample`

  sample size of the data. If there are more than one replicate for each sample, nSample represents the sample size multiplied by the number of replicates.

- `nRNA`
  the number of genes in the data.
- `n-miRNA`
  the number of miRNAs in the data.
- `groupSize`
  the number of samples in either group.
- `permutationTimes`
  the times of permutation used to obtain the empirical distribution of T-stat.
- `round(.)`
  the round function in R. The value of `round(x)` is the closest integer of x.
- `ceiling(.)`
  the ceiling function in R. The value of `ceiling(x)` is the integer part of x added by 1.
- `ncol(.)`
  the ncol function in R. The value of `ncol(data)` is the number of columns of data.
- `mod(.)`
  the mod function in R. The value of `mod(x)` is the remainder of x.

Author(s)

Chen Chu

Maintainer: Chen Chu <chuchenblueblues@gmail.com>

References

deGPS: a Powerful and Flexible Framework for Detecting Differential Expression in RNA-Sequencing Studies

Examples

```
## Not run:
### Analyze real RNA-seq data
### Compare "Early Embryo" to "Late Embryo" in fly data
data(flyData)

i <- 1

group <- rep(1:2, each = 6)
simuData <- as.matrix(flyData$data[ , -1])
simuData <- simuData[ , flyData$compIdx[[i]]]


###remove genes of all-zero read counts
simuData <- simuData[apply(simuData, 1, function(x) !all(x == 0)), ]

### apply deGPS on fly data with the empirical T stats downloaded at
### https://www.dropbox.com/s/if5ido5vd8rzff5/empirical.T.stats.fly.RData
### you can set empirical.T.stats as NULL to get your own empirical values
### note that for non-parallelized computation, it may take hours.

load("empirical.T.stats.fly.RData")
empirical.T.stats.fly <- empTAll[i]
```

```
names(empirical.T.stats.fly) <- "GP-Theta"

### Make sure you have 6 cores before running deGPS_mRNA with nSubcore = ncore = 6
flyRes <- deGPS_mRNA(data = simuData, group = group,
ncore = 6, nSubcore = 6, method = "GP-Theta",
empirical.T.stats = empirical.T.stats.fly)

pvalue <- as.vector(flyRes$pvalue)
adj.p <- p.adjust(pvalue, method = "BH")
topTags(pvalue)

### Generate Random samples from GP(theta, lambda)
examData <- newExampleData(nRNA = 100, groupSize = 6, lambda = 0.9,
theta = 3, ptol = 1e-15)
str(examData)

### Differential Expression Tests
examRes <- deGPS_mRNA(data = examData$data, group = examData$group,
method = "GP-Theta", nSubcore = 2, ncore = 2, geneid = paste("G", 1:100, sep = ""))
str(examRes)

### Generate simulated RNA-seq data from compcodeR package
require(compcodeR)

samples.per.cond <- 5
random.outlier.high.prob <- 0.1
n.vars <- 10000

examData <- generateSyntheticData(dataset = "simuData",
n.vars = n.vars, samples.per.cond = samples.per.cond, n.diffexp = floor(n.vars * 0.1),
repl.id = 1, seqdepth = 1e+07, fraction.upregulated = 0.5,
between.group.diffdisp = FALSE, filter.threshold.total = 1,
filter.threshold.mediancpm = 0, fraction.non.overdispersed = 0,
random.outlier.high.prob = random.outlier.high.prob,
output.file = "simuData_repl1.rds")

group <- examData@sample.annotations$condition

### Make sure you have 6 cores before running deGPS_mRNA with nSubcore = ncore = 6
examRes <- deGPS_mRNA(data = examData@count.matrix, group = group,
method = "GP-Theta", nSubcore = 6, ncore = 6, geneid = paste("G", 1:nrow(examData@count.matrix), sep = ""))
str(examRes)


## End(Not run)
```

deGPS_mRNA *Normalization and Two-group Differential Expression Test for mRNA Read Count Data*

Description

Normalization and Two-group Differential Expression Test for mRNA Read Count Data

Usage

```
deGPS_mRNA(data, dataNormal = NULL, empirical.T.stats = NULL,
group = rep(1:2, each = 5), method = "GP-Theta", nSubcore = 4,
ncore = 4, paired = FALSE, maxIter = 150, geneid = NULL)
```

Arguments

`data` a matrix containing mRNA gene-level read count data. The column represents samples while the row represents genes. Biologcial or technical replicates must be made as new columns in the data.

`dataNormal` not used anymore.

`empirical.T.stats`
A list of empirical T statistics with names as normalization methods, the format of which must be the same as the `empirical.T.stats` in the returned list of `deGPS_mRNA` or `GPSmle`, i.e., a list of empirical T-stats with the method names as the list name. If null, the empirical T statistics will be calculated. Otherwise, only p values are calculated using given empirical T stats.

`group` The group indicator. The length of group must equal to ncol(data).

`method` the methods of normalization. It can be a single charactor or a charactor vector, the values can be "Lowess", "GP-Quantile", "Quantile", "TMM" or "GP-Theta"
.

`nSubcore` The parallel computation strategy splits rows of the data, i.e. mRNAs, into `nSubcore` parts. The empirical T-stats are calculated for each part of the data, and so are the p values. The total number of cores needed in the computation is `nSubcore` * `nMethod`.

`ncore` The total cpu cores used for the calculations. The specified `ncore` can be less than the total number of cores needed (i.e. `nSubcore` * `nMethod`), in which case the cores will be used repeatedly. Apparently, the maximum utilization of the `ncore` cores is reached when `mod(nSubcore * nMethod / ncore) == 0`. You may just make `ncore = nSubcore` to ensure the efficiency of the parallel computation.

`paired` The current version of deGPS only contain unpaired test.

`maxIter` The default value of maxIter is 150. When sample size is large, instead of transversing every possible permutation, randomly shuffling is applied for `maxIter` times to obtain the empirical distributions. Larger `maxIter` costs longer run time. Note that `maxIter` is forced to be not larger than `permutationTimes`.

`geneid` Gene id of the specified data. Biological or technical replicates must be new columns in the data, i.e., duplicates in `geneid` are not allowed.

Details

This function is to analysis mRNA gene-level read count data in two steps: nomalization and permutation based differential expression test.

More than one normalization method can be specified in one run, method `GP-Theta` is suggested.

In permutation DE test, p values are calculated according to the empirical T statistics obtained by randomly shuffling the samples. You can also specify your own empirical T-stats (or the one you get from another run of the function, which is useful in real data based simulations) in argument `empirical.T.stats`.

To deal with the burden of computation, e.g. when SampleSize > 10, parallel computing is embeded in the function. By specify `nSubcore` and `ncore`, parallel computation is applied in the calculations. The abundant genes are divided into `nSubcore` subsets, for each of which the empirical T stats are therefore calculated parallelly. The calculation of p values are also parallelly applied on the subsets of mRNAs using empirical T stats obtained by binding `nSubcore` subsets. `nSubcore * nMethod` cores are needed in total, where `nMethod` represents the number of applied methods.

If the number of cores needed in parallel computing process is larger than `ncore`, cores are iteratively used by the introduced R function `foreach` (Windows) or `mcapply` (Linux). The maximum utilization of the `ncore` cores is reached when `mod(nSubcore * nMethod / ncore) == 0`.

Besides of specifying large `nSubcore` and `ncore`, specify smaller `maxIter` can be also useful to make the function more efficiency. Note that `maxIter` should not be too small, where the empirical distribution may not be reliable.

Value

A `GPSmle` object is returned.

`normalized.data` A list of normalized data, each element represents one speicified normalization method.

`log2FoldChange` The logrithm of fold change of original data.

`empirical.T.stats` A list of the empirical T-stats of normalied data of different normalization methods, generated by permutation of samples. The length of the T-stats is `nRNA * min(permuationTimes, maxIter)`. $permutationTimes = \left(\begin{array}{c} nSample \\ nSample/2 \end{array}\right)/2$, if each group has equal size. $permutationTimes = \left(\begin{array}{c} nSample \\ groupSize \end{array}\right)$, if each group has unequal size.

`log2FoldChange` The logarithm of fold change of original data.

`pvalue` The resulting pvalues. Note that the pvalues may be slightly different in different runs of deGPS for the same data when not all possible permutations are transversed in the calculations of empirical T stats.

`paired` FALSE

`method` the normalization methods applied to get the result.

`type` "mRNA"

Author(s)

Chen Chu

See Also

`GPSmle`

Examples

```
## Not run:
### See the example in "flyData" for real data analysis and the comparison between deGPS
### and other widely-used methods
```

```
##Generate Random samples from GP(theta, lambda)
examData <- newExampleData(nRNA = 100, groupSize = 6, lambda = 0.9,
theta = 3, ptol = 1e-15)
str(examData)

##Differential Expression Tests
examRes <- deGPS_mRNA(data = examData$data, group = examData$group,
method = "GP-Theta", nSubcore = 2, ncore = 2, geneid = paste("G", 1:100, sep = ""))
str(examRes)
topTags(examRes, n = 10, method = "BH")

###Generate simulated RNA-seq data from compcodeR package
require(compcodeR)

samples.per.cond <- 5
random.outlier.high.prob <- 0.1
n.vars <- 10000

examData <- generateSyntheticData(dataset = "simuData",
n.vars = n.vars, samples.per.cond = samples.per.cond, n.diffexp = floor(n.vars * 0.1),
repl.id = 1, seqdepth = 1e+07, fraction.upregulated = 0.5,
between.group.diffdisp = FALSE, filter.threshold.total = 1,
filter.threshold.mediancpm = 0, fraction.non.overdispersed = 0,
random.outlier.high.prob = random.outlier.high.prob,
output.file = "simuData_repl1.rds")

group <- examData@sample.annotations$condition

###Make sure you have 6 cores before running deGPS_mRNA with nSubcore = ncore = 6
examRes <- deGPS_mRNA(data = examData@count.matrix, group = group,
method = "GP-Theta", nSubcore = 6, ncore = 6, geneid = paste("G", 1:nrow(examData@count.matrix), sep = ""))
str(examRes)
topTags(examRes, n = 10, method = "BH")


## End(Not run)
```

flyData *A real RNA-seq data set of fly*

Description

A real RNA-seq data set of fly

Usage

```
data(flyData)
```

Details

A RNA-seq data set of fly downloaded at Fly Data, study `modencodefly`

References

The developmental transcriptome of Drosophila melanogaster, Graveley BR and etc., Nature 2011 Mar 24;471(7339):473-9.

Examples

```
## Not run:
### load required packages

require(edgeR)
require(DESeq)
require(DESeq2)
require(ggplot2)
require(gridExtra)
require(deGPS)

data(flyData)
str(flyData)

################################################################
#### The list of flyData contains:
#### data: read counts table with the first row as gene names
#### groupInfo: the state name of each sample in the data
#### compIdx: six subgroups of indices of samples for analysis
################################################################
#### choose the i-th subgroup as an example:
#### i = 1: Early vs Late Embryo
#### i = 2: Late Embryo vs Larval
#### i = 3: Larval vs Adult
#### i = 4: Early Embryo vs Larval
#### i = 5: Early Embryo vs Adult
#### i = 6: Late Embryo vs Adult
################################################################

i <- 1

group <- rep(1:2, each = 6)
simuData <- as.matrix(flyData$data[ , -1])
simuData <- simuData[ , flyData$compIdx[[i]]]
titleName <- names(flyData$compIdx)[i]            ### the name used in the title of the final plot

###remove genes of all-zero read counts
simuData <- simuData[apply(simuData, 1, function(x) !all(x == 0)), ]

### apply deGPS on fly data with the empirical T stats downloaded at
### https://www.dropbox.com/s/if5ido5vd8rzff5/empirical.T.stats.fly.RData
### you can set empirical.T.stats as NULL to get your own empirical values
### note that for non-parallelized computation, it may take hours.

load("empirical.T.stats.fly.RData")
empirical.T.stats.fly <- empTAll[i]
names(empirical.T.stats.fly) <- "GP-Theta"

### Make sure you have 6 cores before running deGPS_mRNA with nSubcore = ncore = 6
flyRes <- deGPS_mRNA(data = simuData, group = group,
ncore = 6, nSubcore = 6, method = "GP-Theta",
empirical.T.stats = empirical.T.stats.fly)

pvalue <- as.vector(flyRes$pvalue)

### compare result with edgeR and DESeq, DESeq2

d0 <- DGEList(counts = simuData, group = group)
design <- model.matrix(~ group, data = d0$samples)
d <- try(calcNormFactors(d0, method = "TMM"))
d <- try(estimateGLMCommonDisp(d, design, verbose = TRUE))
d <- try(estimateGLMTrendedDisp(d, design))
efit <- try(glmQLFTest(d, design, coef = 2))
edge2Res <- efit$table$PValue

d <- try(estimateGLMTagwiseDisp(d, design))
efit <- try(glmFit(d, design))
efit1 <- try(glmLRT(efit, coef = 2))
edge1Res <- efit1$table$PValue

cds1 <- newCountDataSet(simuData, group)
cds2 <- estimateSizeFactors(cds1)
cds3 <- try(estimateDispersions(cds2))
if("try-error"
if("try-error"
fitType = "local"))
res <- nbinomTest(cds3, 1, 2)
deseqRes <- res$pval
deseqRes[is.na(deseqRes)] <- 1

dds <- DESeqDataSetFromMatrix(countData = simuData,
colData = data.frame(group = group),
design = ~ group)
dds <- DESeq(dds)
res <- results(dds)
deseq2Res <- res$pvalue

### plot the overlap of DEs
pAll <- cbind(pvalue, edge1Res, edge2Res, deseqRes, deseq2Res)

pAll[is.na(pAll)] <- 1

pAll <- apply(pAll, 2, p.adjust, method = "BH")

overLapTemp <- overLapTemp1 <- matrix(NA, ncol(pAll), ncol(pAll))

for(ii in 1:ncol(pAll)){
for(jj in 1:ncol(pAll)){
overLapTemp[ii, jj] <- sum(pAll[ , ii] < 0.05 & pAll[ , jj] < 0.05) / sum(pAll[ , ii] < 0.05)
}
}

for(ii in 1:ncol(pAll)){
for(jj in 1:ncol(pAll)){
overLapTemp1[ii, jj] <- sum(pAll[ , ii] < 0.05 & pAll[, jj] < 0.05)
}
}
```

```
compName <- c("deGPS", "edgeR1", "edgeR2", "DESeq", "DESeq2")
dimnames(overLapTemp) <- dimnames(overLapTemp1) <- list(compName, compName)

jpeg("flyOverLap.jpg", width = 1600, height = 700)

a <- levelplot(overLapTemp, xlab = "", ylab = "", main = list(paste("Overlap Proportion", titleName), cex =
scale = list(cex = 1.3),
colorkey = list(labels = list(cex = 1.2)),
panel=function(...) {
arg <- list(...)
panel.levelplot(...)
panel.text(rep(1:nrow(overLapTemp), ncol(overLapTemp)),
rep(1:ncol(overLapTemp), each = nrow(overLapTemp)), round(as.vector(overLapTemp), 2), cex = 1.5)}
)

b <- levelplot(overLapTemp1, xlab = "", ylab = "", main = list(paste("Overlap Number", titleName), cex = 2
scale = list(cex = 1.3),
colorkey = list(labels = list(cex = 1.2)),
panel=function(...) {
arg <- list(...)
panel.levelplot(...)
panel.text(rep(1:nrow(overLapTemp1), ncol(overLapTemp1)),
rep(1:ncol(overLapTemp1), each = nrow(overLapTemp1)), as.vector(overLapTemp1), cex = 1.5)}
)
grid.arrange(a, b, ncol=2)
dev.off()

## End(Not run)
```

GPSmle GPSmle.default

Description

GPSmle.default

Usage

```
GPSmle(data, group = rep(1:2, each = 5),
type = c("pvalue", "normalization", "ecdf"),
method = c("GP-Theta", "Lowess", "GP-Quantile", "Quantile", "TMM"),
maxIter = 500, paired = FALSE, ncpu = 1, geneid = NULL, empirical.T.stats = NULL)
```

Arguments

data	GPSmle.default
group	GPSmle.default
type	GPSmle.default
method	GPSmle.default
maxIter	GPSmle.default
paired	GPSmle.default
ncpu	GPSmle.default

`geneid` GPSmle.default

`empirical.T.stats` GPSmle.default

Details

GPSmle.default

See Also

GPSmle.default

`GPSmle.default` *Generalized Poisson Statistical Maximum Likelihood Estimation (default)*

Description

the default method for the function `GPSmle`.

Usage

```
## Default S3 method:
GPSmle(data, group = rep(1:2, each = 5),
type = c("pvalue", "normalization", "ecdf"),
method = c("GP-Theta", "Lowess", "GP-Quantile", "Quantile", "TMM"),
maxIter = 500, paired = FALSE, ncpu = 1, geneid = NULL, empirical.T.stats = NULL)
```

Arguments

`data` a matrix containing microRNA read count data. The column represents samples while the row represents miRNAs. Biologcial or technical replicates must be made as new columns in the data.

`group` The group indicator. The length of group must equal to ncol(data).

`type` `type` can be "normalization", "ecdf" or "pvalue", to which step `GPSmle` will stop. "normalization" means that only normalized data is returned and no DE test is conducted; "ecdf" means the empirical T-stats are generated after nomalization, and the output contains both the normalized data sets and empirical values; "pvalue" means p-values are calculated after the empirical T-stats are obtained, and the output contains the normalized data sets, empirical T-stats and the p-values of DE test.

`method` The methods of normalization. More than one method can be specified. The value can be "Lowess", "Quantile", "TMM", "GP-Quantile", "GP-Theta" or "GP-MLE2L". See the reference for more details.

`maxIter` The default value of maxIter is 500. When sample size is large, instead of transversing every possible permutations, randomly sampling is applied for `maxIter` to obtain the empirical distributions. Larger `maxIter` costs longer run time. Note that `maxIter` is forced to be not larger than `permutationTimes`.

`paired` The current version of deGPS only contain unpaired test.

ncpu
: The number of cores for the parallel computing. When sample size is large, the permutation step may be time-consuming. Specify ncpu > 1, parallel computation is applied in the function. ncpu cores are used to calculate maxIter times of permutations, each of which take responsibilies of part of the permutation task. The calculation of p values are also splitted into subsets for parallel computation.

geneid
: Gene id of the specified data. Biological or technical replicates must be new columns in the data, i.e., duplicates in geneid are not allowed.

empirical.T.stats
: A list of empirical T statistics with names as normalization methods, the format of which must be the same as the empirical.T.stats in the returned list of deGPS_mRNA or GPSmle, i.e., a list of empirical T-stats with the method names as the list name. If null, the empirical T statistics will be calculated. Otherwise, only p values are calculated using given empirical T stats.

Details

This function is to analyze miRNA read count data in two steps: normalization and two-group differential expression test.

More than one normalization method can be specified in method when ncpu = 1. Method GP-Theta is suggested. There are also other choices of the normalization methods. More details about GP-Quantile, GP-Theta can be found in our article.

When sample size is large, maxIter must be specified (500 by default). Smaller maxIter may save run time but to be too small may make the empirical distribution unreliable.

Besides of specifying appropriately small value of maxIter, it is suggested to make ncpu larger than 1, where parallel computation is applied. In parallel computing process, permutation task is splitted into parts of almost equal size, each of which will be processed by a core. And the calculation of p values is also paralleled by dividing the miRs into subsets, each of which is processed by a core.

Value

A GPSmle object. See deGPS_mRNA.

References

deGPS: a Powerful and Flexible Framework for Detecting Differential Expression in RNA-Sequencing Studies

See Also

deGPS_mRNA

Examples

```
## Not run:
##Generate Random samples from GP(theta, lambda)
examData <- newExampleData(nRNA = 100, groupSize = 2, lambda = 0.9,
theta = 3, ptol = 1e-15)
str(examData)

##Differential Expression Tests for miRNA
examRes <- GPSmle(data = examData$data, group = examData$group, method = "GP-Theta",
type = "pvalue", ncpu = 1, geneid = paste("G", 1:100, sep = ""))
```

```
str(examRes)

topTags(examRes, n = 10, method = "BH")

plot(examRes)

## End(Not run)
```

GPSmleEst GPSmle.default

Description

GPSmle.default

Usage

```
GPSmleEst(data, group = rep(1:2, each = 5),
type = c("normalization", "ecdf", "pvalue"),
dataNormal = NULL, empirical.T.stats = NULL,
method = c("Lowess", "GP", "Quantile", "TMM", "GP2"),
maxIter = 500, paired = FALSE, ncpu = 1, geneid = NULL)
```

Arguments

data	GPSmle.default
group	GPSmle.default
type	GPSmle.default
dataNormal	no longer validated
method	GPSmle.default
maxIter	GPSmle.default
paired	GPSmle.default
ncpu	GPSmle.default
geneid	GPSmle.default
empirical.T.stats	GPSmle.default

Details

GPSmle.default

See Also

GPSmle.default

newExampleData *Generate example data for GPSmle and deGPS_mRNA*

Description

Randomly generate Generalized Poisson distributed samples with given theta and lambda.

Usage

```
newExampleData(nRNA = 100, groupSize = 5, lambda = 0, theta = 1, ptol = 1e-10)
```

Arguments

`nRNA`	The number of genes or miRs.
`groupSize`	A integer represents The number of samples in each group. Note that the function can only generate two equal gropus.
`lambda`	The lambda parameter of GP distribution. The values must be within (0, 1). Since miRNA/mRNA read counts tend to be overdispersed, we constain the lambda of example data larger than zero to be similar to the real cases. Note that the length of `lambda` can be either one or two, representing the equal or unequal lambda for each group.
`theta`	The theta parameter of GP distribution. Must be positive values. It can be a single value or a numeric vector with length two.
`ptol`	The tolerance of probabilites of GP distribution. The default value is 1e-15, since regular R can not tell the difference smaller than 1e-15. See `details` for more explanations.

Details

The resulting data set contains two GP distributed groups with the specified lambda and theta as the parameters, with each group containing `groupSize` samples. Note that the length of `lambda` and `theta` can be either one or two, representing the same or different GP for two groups. Moreover, the data is neither H0 (non-DE) nor H1 (with DE) data. Every element is a random sample of the given GP and one can not tell whether one single row in the data is DE. However, with large amount of RNAs in the data, it tends to be H0 data, since the variability in one particular row is caused by random assignment of two GP distributions.

The random samples of the specified GP distribution are generated from a multinomial distribution, the domain of which is from zero to a maximum value – `gpMax` + 1. The larger `gpMax` is, the closer two distributions are.

The maximum integer `gpMax` is determined as the minimum integer satisfying $P(x = gpMax) \geq ptol$. The probability for each value from zero to gpMax + 1 is then calculated as the probability of that in specified GP distribution. Note that the probability of $P(x = gpMax + 1) = 1 - P(x = 0) - \ldots - P(x = gpMax)$. Hence, the smaller `ptol` is, the closer the approximated multinomial distribution is to the specified GP distribution. And since regular R, i.e. without particular package which enables more precise calculations, can not tell differences smaller than 1e-15, `ptol` is set as 1e-15.

Another way to generate simulated RNA-seq data is the `compcodeR` package. Details can be found in the package manual and user guide document. See a simple example in the following example session.

Value

group	The group indicator of the resulting data.
data	The resulting data with GP distributon.

See Also

deGPS_mRNA, GPSmle

Examples

```
## Not run:
####Different Lambda and Theta for Two Groups
examData <- newExampleData(nRNA = 100, groupSize = 2, lambda = c(0.5, 0.9),
theta = c(3, 10), ptol = 1e-15)

####Same Lambda and Theta for Two Groups
examData <- newExampleData(nRNA = 100, groupSize = 2, lambda = 0.9, theta = 3,
ptol = 1e-15)


###Generate simulated RNA-seq data from compcodeR package
require(compcodeR)

samples.per.cond <- 5
random.outlier.high.prob <- 0.1
n.vars <- 10000

examData <- generateSyntheticData(dataset = "simuData",
n.vars = n.vars, samples.per.cond = samples.per.cond, n.diffexp = floor(n.vars * 0.1),
repl.id = 1, seqdepth = 1e+07, fraction.upregulated = 0.5,
between.group.diffdisp = FALSE, filter.threshold.total = 1,
filter.threshold.mediancpm = 0, fraction.non.overdispersed = 0,
random.outlier.high.prob = random.outlier.high.prob,
output.file = "simuData_repl1.rds")

## End(Not run)
```

plot.GPSmle *Plot GPSmle*

Description

Plot the histograms of GPSmle results.

Usage

```
## S3 method for class GPSmle
plot(x, ...)
```

Arguments

x	the object returned by GPSmle.
...	the parameters of plot

Details

See `GPSmle.default` for more details of the output of `GPSmle`.

Value

The output depends on the specification of `type` in `GPSmle`. If type = "normalization", the histograms of normalized data sets are returned. So are the "ecdf" and "pvalue" or "mRNA" in `deGPS_mRNA`.

`summary.GPSmle` *Summary of GPSmle*

Description

Summary of GPSmle

Usage

```
## S3 method for class GPSmle
summary(object, ...)
```

Arguments

`object`	the GPSmle object returned by `GPSmle` or `deGPS_mRNA`.
`...`	see the `summary.default`

Details

summary of the GPSmle

Value

summary of the GPSmle

`topTags` *The top significant genes or miRs*

Description

The top significant genes or miRs

Usage

```
topTags(x, n = 10, method = "BH", significance = 0.05)
```

Arguments

`x` A `GPSmle` object returned by `GPSmle` or `deGPS_mRNA` or a p value vector. If it is a `GPSmle` object, only one column of p values is allowed in this function.

`n` the number of required top significant genes or miRs.

`method` the adjust method of multiple testing p values, same as R function `p.adjust`.

`significance` the significant level of the DE.

Details

This function is to find the significant genes or miRs at the given significant level. If you want to call this function, make sure that `library(deGPS)` is called after other packages, such as `edgeR`, since those packages also contain function named `topTags`.

Value

A list containing:

`pvalue` the ordered p values of significant genes or miRs.

`adj.pvalue` the ordered adjusted p values by specified method.

`method` p value adjusted method, same as the method in R function `p.adjust`

`geneName` names of the genes or miRs.

`geneid` the row indice of the genes or miRs in given p values.

`significance` the specified significant level.

Index