Package ‘PROPER’

November 15, 2025

Type Package

Title PROspective Power Evaluation for RNAseq

Version 1.43.0

Imports edgeR

Depends R (>= 3.3)

Suggests BiocStyle,DESeq2,DSS,knitr

Date 2020-11-25

Author Hao Wu

Maintainer Hao Wu <hao.wu@emory.edu>

Description This package provide simulation based methods for evaluating the statistical power in differential expression analysis from RNA-seq data.

License GPL

biocViews ImmunoOncology, Sequencing, RNASeq, DifferentialExpression

VignetteBuilder knitr

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

git_branch devel

git_last_commit 54779f4

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-11-14

Contents

comparePower 2
estParam 4
param 6
plotPowerHist 7
plots 8
power.seqDepth 9

RNAseq.SimOptions.2grp 10
runSims 12
simRNAseq 14
summaryPower 15

Index **17**

`comparePower` *Compute the power-related quantities from simulation results*

Description

This function take the simulation output from "runSims" function and compute a variety of power-related quantities.

Usage

```
comparePower(simOutput, alpha.type = c("fdr", "pval"), alpha.nominal=0.1,
             stratify.by = c("expr", "dispersion"), strata,
             filter.by = c("none","expr"), strata.filtered = 1,
             target.by=c("lfc", "effectsize"), delta=0.5)
```

Arguments

`simOutput`	The result from "runSims" function.
`alpha.type`	A string to represent the way to call DE genes. Available options are "fdr" and "pval", for calling DE genes based on FDR or p-values. Default is "fdr".
`alpha.nominal`	The nomial value for call DE genes. Recommend values are 0.1 when using FDR, and 0.01 for using p-values.
`stratify.by`	A string to represent the way to stratify genes. Available options are "expr" and "dispersion", for stratifying gene by average expression levels or over-dispersion.
`strata`	The strata used for stratification.
`filter.by`	A string to represent the way to filter genes. This is used in conjunction with strata.filtered for gene filtering. Available options are "none" and "expr". "none" stands for no filtering, thus all genes will be considered. "expr" stands for filtering based on average expression levels.
`strata.filtered`	The strata to be filtered out in computing power-related quantities. Genes fall into these strata will be excluded when computing power-related quantities. See "Details" for more description of gene filtering.
`target.by`	A string to specify the method to define "biologically important" DE genes. Available options are (1) "lfc": interesting genes are defined by absolute log fold changes. (2) "effectsize": interesting genes are defined by absolute log fold changes divided by the square root of dispersion.
`delta`	A threshold used for defining "biologically important" genes. Genes with absolute log fold changes (when target.by is "lfc") or effect sizes (when target.by is "effectsize") greater than this value are deemed DE in power calculations. See "Details" for more description.

Details

This is the main function to compute various power-related quantities, under stratification and filtering of all genes.

Gene stratification: we advocate to compute and visualize the powers at different stratification of genes. Because of the characteristics of RNA-seq data such as average expression level (counts), powers for calling DE genes are different even when the magnitude of changes are the same. The stratified results will provide a more comprehensive power assessment and better guide the investigators in experimental designs and analysis strategies.

Gene filtering: sometimes it is advisible to filter out some genes (such as the ones with very low count) before DE detection. The filtering option here provides an opportunity to compare the powers before and after filtering.

Define biologically interesting genes: we advocate to compute powers for "biologically interesting genes", because there might be many true DE genes with very low changes which are difficult to detect. In this sense, we are only interested in genes with adequate changes between groups (defined as "biologically interesting"). We provide two options to define biologically interesting genes: by absolute values of log fold changes or effect sizes (absolute values of log fold changes divided by the square root of dispersions). Genes with these quantities over a threshold are deemed intersting, and the power calculation are based on these genes.

Value

A list with following fields:

`TD`, `FD`, `alpha`, `FDR`, `power`
: 3D array representing the number of true discoveries, false discoveries, type I error rate, FDR, and power for all simulation settings. The dimension of the arrays are nstrata * N * nsims. Here nstrata is number of specified strata. N is number of different sample sizes settings, and nsims is number of simulations.

`alpha.marginal`, `FDR.marginal`, `power.marginal`
: Matrix representing the marginal type I error rate, FDR, and power for all simulation settings. The dimension of the matrices are N * nsims. These are the marginalized (over all strata) values of the stratified quantities.

`stratify.by`
: The input stratify.by.

`strata`
: The input strata.

`Nreps`
: Sample sizes one wants to perform simulation on. This is taken from the simulation options.

`target.by`
: The input method to define "biologically important" DE genes.

`delta`
: The input delta for biologically important genes.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

RNAseq.SimOptions.2grp, simRNAseq, runSims

Examples

```
## Not run:
simOptions = RNAseq.SimOptions.2grp()
## run a few simulations
simRes = runSims(Nreps=c(3,5,7), sim.opts=simOptions, nsims=5,
                 DEmethod="edgeR")

## using FDR 0.1 to call DE, then look at power curves and summary
powers = comparePower(simRes)
summaryPower(powers)
par(mfrow=c(2,2))
plotPower(powers)
plotPowerTD(powers)
plotFDR(powers)
plotFDcost(powers)

## filter out the genes with low counts (<10) and redo power calculation
## Marginal powers are significantly higher.
powers = comparePower(simRes, filter.by="expr", strata.filtered=1)
summaryPower(powers)
par(mfrow=c(2,2))
plotPower(powers)
plotPowerTD(powers)
plotFDR(powers)
plotFDcost(powers)

## Provide higher threshold for log fold change to define true DE.
## This will result in higher power.
powers2 = comparePower(simRes, delta=2)
summaryPower(powers2)
par(mfrow=c(2,2))
plotPower(powers2)
plotPowerTD(powers2)
plotFDR(powers2)
plotFDcost(powers2)

## use effect size to define biologically interesting genes
powers3 = comparePower(simRes, filter.by="expr", strata.filtered=1,
                       target.by="effectsize", delta=1)
summaryPower(powers3)
par(mfrow=c(2,2))
plotPower(powers3)
plotPowerTD(powers3)
plotFDR(powers3)
plotFDcost(powers3)

## End(Not run)
```

`estParam` *estimate simulation parameters from a count table*

Description

This function estimate simulation parameters from a count table, which can be used to set simulation scenarios. The parameters estimated include baseline expression, biological variation in the form of dispersion paramter.

Usage

```
estParam(X,type = c(1,2))
```

Arguments

`X`	An ExpressionSet or a matrix, with entries being RNA sequencing counts
`type`	`type=1` estimates with minimal assumption based on sample means and variances for each genes. `type=2` uses `edgeR` package.

Details

This is a function that allows a user to establish simulation basis from his/her own data set. Type 1 uses simple average count and sample variance to estimate over dispersion, after normalizing by library size for each sample. Estimated dispersion is bounded at minimum 0.001. Type 2 uses `edgeR`'s `AveLogCPM` as the estiamte for baseline expression and `tagwiseDispersion` as the estimate for dispersion.

Value

A list with following fields:

`seqDepth`	Estimated sequencing depth for each sample
`lmean`	A vector of baseline expression rate per gene in log scale, with length `nrow(X)`
`lOD`	A vector of log dispersion for each gene, with length `nrow(X)`

Author(s)

Jean Wu <zhijin_wu@brown.edu>

Examples

```
## Not run:
lmu0=rnorm(20000,5,2)
lOD0=rnorm(20000,-4,1)
lmu=exp(rnorm(20000*10,lmu0,exp(lOD0)))
X=matrix(rpois(20000*10,lmu),20000,10)
param=estParam(X)

## End(Not run)
```

param *Some existing RNA-seq and gene expression microarray data*

Description

These are several datasets distributed with the package. They are simulation parameters estimated from existing RNA-seq data. The original raw count data were obtained from reCount website.

Usage

```
data(cheung)
data(gilad)
data(bottomly)
data(maqc)
data(GE.human)
data(pbmc)
```

Details

The data include:

- cheung: parameters from Cheung data. They are measurements from unrelated individuals, so the dispersions are large.
- gilad: parameters from Gilad data. They are for Human liver sample comparisons between male and female. This dataset has moderate dispersions.
- bottomly: parameters from Bottomly data. They compare two strains of inbred mice and the within group dispersions are small.
- maqc: MAQC data which are technical replicates. There are no biological variation from the replicates, so the dispersions are close to 0.
- GE.human: a vector of aggregated counts from all samples in Cheung data. This is used for generating marginal expression distribution when provide sequnencing depth.
- pbmc: data from gene expression microarray data. This is for an example of using historical data to estimate effect sizes.

Author(s)

Hao Wu <hao.wu@emory.edu>

plotPowerHist — *Plot the histogram of power*

Description

Using histogram-alike graph to visualize the distribution of all genes and DE genes in all user specified strata.

Usage

```
plotPowerHist(powerOutput, simResult, main = "Histogram of power", return = FALSE)
```

Arguments

powerOutput	Result object from "comparePower" function.
simResult	Result object from "runSims" function.
main	Figure caption.
return	Return a matrix for average number of genes and true DE genes in each strata.

Value

A matrix of two rows. First row is for number of genes, and the second row is for number of DE genes. Columns are strata.

Author(s)

Hao Wu <hao.wu@emory.edu>

Examples

```
## Not run:
simOptions = RNAseq.SimOptions.2grp()
## run a few simulations
simRes = runSims(Nreps=c(3,5,7), sim.opts=simOptions, nsims=5,
                 DEmethod="edgeR")

## using FDR 0.1 to call DE, then look at power curves and summary
powers = comparePower(simRes)
plotPowerHist(powers, simRes)

## End(Not run)
```

plots *Plot the stratified curves for power-related quantities*

Description

These are a group of functions to generate plot to visualize the stratified power metrics, including stratified power, FDR, true discoveries, false discoveries, false discovery cost, and type I error. There will be a line for each sample size. X-axis are strata and y-axis are the metrics. Error bars will be plotted if requested.

Usage

```
plotPower(powerOutput, cols = 1:ncol(powerOutput$FD),
    lty = 1:ncol(powerOutput$power), main = "", ylab = "Power",
    leg = TRUE, error.bar=TRUE)

plotFDR(powerOutput, cols = 1:ncol(powerOutput$FDR), lty=1:ncol(powerOutput$FDR),
    main = "", ylab="FDR", leg = TRUE, error.bar=TRUE)

plotPowerTD(powerOutput, cols = 1:ncol(powerOutput$TD),
    lty=1:ncol(powerOutput$TD), main="", ylab = "# True Discoveries",
    leg = TRUE, error.bar=TRUE)

plotPowerFD(powerOutput, cols = 1:ncol(powerOutput$FD), lty=1:ncol(powerOutput$FD),
    main = "", ylab = "# False Discoverie", leg = TRUE, error.bar=TRUE)

plotFDcost(powerOutput, cols = 1:ncol(powerOutput$FD), lty=1:ncol(powerOutput$FD),
    main="", ylab = "False discovery cost", leg = TRUE, error.bar=TRUE)

plotPowerAlpha(powerOutput, cols = 1:ncol(powerOutput$alpha), lty=1:ncol(powerOutput$alpha),
    main = "", ylab = "False positive rate", leg = TRUE, error.bar=TRUE)

plotAll(powerOutput)
```

Arguments

`powerOutput`	Result object from "comparePower" function.
`cols`, `lty`	Colors and line types of the TD curves.
`main`	Figure title.
`ylab`	Y-axis label.
`leg`	Indicator for having figure legend or not.
`error.bar`	Indicator for whether plotting error bars in the figures.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

comparePower

Examples

```
## Not run:
simOptions = RNAseq.SimOptions.2grp()
## run a few simulations
simRes = runSims(Nreps=c(3,5,7), sim.opts=simOptions, nsims=5,
                 DEmethod="edgeR")

## using FDR 0.1 to call DE, then look at power curves and summary
powers = comparePower(simRes)
## plot
par(mfrow=c(2,3))
plotPower(powers)
plotPowerTD(powers)
plotFDR(powers)
plotFDcost(powers)
plotPowerAlpha(powers)

## in one figure
par(mfrow=c(2,3))
plotAll(powers)

## End(Not run)
```

---

power.seqDepth *Estimate the marginal power under different sequencing depth*

Description

The function helps study the power under different sequencing depth and sample sizes. It estimates the marginal powers based on existing simulation results, given new sequencing depth.

Usage

```
power.seqDepth(simResult, powerOutput, depth.factor = c(0.2, 0.5, 1, 2, 5, 10))
```

Arguments

simResult	Result object from "runSims" function.
powerOutput	Result object from "comparePower" function.
depth.factor	A vector of numbers specifying the *relative* sequencing depth, comparing to the depth used in the simulation. 1 means using the same number of total reads as the simulation.

Details

The powers under different sequencing depth and sample sizes provides important guidence in experimental design. Under the same total number of sequence reads, investigator can choose to use more replicates and shallower coverage for each, or less replicates and deeper coverage.

This function provides estimated marginal power holding all experimental variables fixed (biological variation, effect sizes, sample sizes, etc.) except the sequencing depth. Changing sequencing depth will only alter the marginal distribution of average counts. Since the stratified power (by average counts) won't change, those numbers are used in estimating the powers under different depth. This approaches allows skipping new simulations, which saves computation.

Value

A matrix for marginal powers. Each row is for a sequencing depth, each columns is for a sample size.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

comparePower, summary.power

Examples

```
## Not run:
simOptions = RNAseq.SimOptions.2grp()
simRes = runSims(Nreps=c(3,5,7), sim.opts=simOptions, nsims=5,
                 DEmethod="edgeR")
powers = comparePower(simRes)
power.seqDepth(simRes, powers)

## End(Not run)
```

RNAseq.SimOptions.2grp

Set up options for simulating RNA-seq data in two-group comparison.

Description

This function takes user provided options for for simulating RNA-seq data, and return a list. The result of this function will be the input for "runSims" and "simRNAseq" function.

Usage

```
RNAseq.SimOptions.2grp(ngenes, seqDepth, lBaselineExpr, lOD, p.DE, lfc, sim.seed)
```

Arguments

`ngenes`	Number of genes in the simulation.
`lBaselineExpr`	log baseline expression for each gene. This can be: (1) a constant; (2) a vector with length ngenes; (3) a function that takes an integer n, and generate a vector of length n; (4) a string specifying the name of existing datasets, from which the mean expressions will be sampled. Details for this option is provide in "Details" section.
`lOD`	log over-dispersion for each gene. Available options are the same as for lBaselineExpr.
`seqDepth`	Sequencing depth, in terms of total read counts. This will be ignored if lBaselineExpr is specified.
`p.DE`	Percentage of genes being differentially expressed (DE). By default it's 5%.
`lfc`	log-fold change for DE genes. This can be: (1) a constant; (2) a vector with length being number of DE genes; (3) a function that takes an integer n, and generate a vector of length n. If the input is a vector and the length is not the number of DE genes, it will be sampled with replacement to generate log-fold change.
`sim.seed`	Simulation seed.

Details

The simulation of RNA-seq data requires a lot of parameters. This function provides users an interface to specify the simulation parameters. The result from this function will be used for simulating RNA-seq count data. By default, the simulation parameters are similar to that from Cheung data (for unrelated individuals, with large biological variance).

The baseline expression levels and log over-dispersions can be sampled from real data. There are parameters estimated from several real datasets distributed with the package. Available string options for "lBaselineExpr" and "lOD" include: (1) "cheung": parameters from Cheung data, which measures unrelated individuals, so the dispersions are large; (2) "gilad": from Gilad data which are for Human liver sample comparisons between male and female. This dataset has moderate dispersions; (3) "bottomly": from Bottmly data which are from comparing two strains of inbred mice. The dispersions are small. (4) "maqc": from MAQC data which are technical replicates. There are no biological variation from the replicates because the data are technical replicates. The dispersions from this dataset is very small.

The effect sizes (log fold changes of the DE genes) are arbitrarily specified. It is possible to estimate those from real data. We provide a simple example in the package vignette for doing so.

Value

A list with following fields:

`ngenes`	An integer for number of genes.
`p.DE`	Percentage of DE genes.
`lBaselineExpr`	A vector of length ngenes for log baseline expression.
`lOD`	A vector of length ngenes for log over-dispersion.

`lfc`	A vector of length (ngenes*p.DE) for log fold change of the DE genes.
`sim.seed`	The specified simulation seed.
`design`	A string representing the experimental design. From this function it is '2grp', standing for two-group comparison.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

simRNAseq,runSims

Examples

```
## default
simOptions=RNAseq.SimOptions.2grp()
summary(simOptions)

## specify some parameters: generate baseline expression and
## dispersion from Bottom data, and specify a function for
## alternative log fold changes.
fun.lfc=function(x) rnorm(x, mean=0, sd=1.5)
simOptions=RNAseq.SimOptions.2grp(ngenes=30000, lBaselineExpr="bottomly",
       lOD="bottomly", p.DE=0.05, lfc=fun.lfc)
summary(simOptions)
```

`runSims` *Run a number of RNA-seq simulations and DE detections*

Description

This is the "wrapper" function for running RNA-seq DE detection simulation. It runs simulations under different sample sizes (replicates in each group) for a certain numbers. In each simulation, the RNA-seq data are generated, and then DE detection (using user specified method/software) is performed. The return object contains DE test results (test statistics, p-values, FDRs, etc.) from all simulations.

Usage

```
runSims(Nreps = c(3, 5, 7, 10), Nreps2, nsims = 100, sim.opts,
        DEmethod = c("edgeR", "DSS", "DESeq2"), verbose =TRUE)
```

Arguments

`Nreps`	Sample sizes one wants to perform simulation on. This is a vector, each element specifies the number of biological replicates in each group. Default value is c(3, 5, 7, 10), which means we want to perform simulation when there are 3, 5, 7, or 10 replicates in each group.
`Nreps2`	Sample sizes for the second treatment group. If this is missing, it'll take the same value as Nreps and then two groups will be assumed to have the same sample size. This parameter allows two treatment groups have different sample sizes. When specified, it must be a vector of the same length as Nreps.
`nsims`	Number of simulations.
`sim.opts`	An object for simulation option. This should be the return object from "RNAseq.SimOptions.2grp" function.
`DEmethod`	String to specify the DE detection method to be used. Available options are "edgeR", "DSS", and "DESeq2".
`verbose`	Logical value to indicate whether to output some messages (progress report).

Details

This is the main simulation function in the packge. After specifying the simulation parameters (from "RNAseq.SimOptions.2grp" function), one wants to evaluate the power vs different sample sizes under that simulation setting. This function simulates the count data and performs statistical tests for DE detection. It only stores and returns the DE test results (test statistics, p-values, FDRs, etc.) but doesn't make inferences. The inferences will be conducted in the "comparePower" function. The advantage is that for one simulation setting, the simulation only need to be run once. The inferences using different critical values and type I error controls can then be drawn from the same results. This greatly save the computation because the simulation part is the most computationally intensive.

This function can be slow, depends on the setting (number of genes, replicates, simulations, etc). For the default (50000 genes, 100 simulations, for 3, 5, 7, or 10 replicates), it takes about an hour to run on a single core of i7 2.7GHz CPU. But again, this only need to be run once for a particular simulation setting.

Value

A list with following fields:

`pvalue`, `fdrs`	3D array for p-values and FDR from each simulation. The dimension of the array is ngenes * N * nsims. Here N is length(Nreps), of the number of different sample sizes settings.
`xbar`	3D array for average read counts for genes. Dimension is the same as pvalue/fdr. This will be used in "comparePower" function for stratified power quantities.
`DEid`	A list of length nsims. Each contains the id of DE genes.
`lfcs`	A list of length nsims. Each contains the log fold changes of DE genes.
`Nreps`	The input Nreps.
`sim.opts`	The input sim.opts.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

RNAseq.SimOptions.2grp, simRNAseq

Examples

```
simOptions = RNAseq.SimOptions.2grp()
## using 3 different sample sizes, run 2 simulations, using edgeR
simRes = runSims(Nreps=c(3,5,7), sim.opts=simOptions, nsims=2,
                 DEmethod="edgeR")
names(simRes)

## Not run:
## using 5 different sample sizes, run 100 simulations, using edgeR.
## This will be slow.
simRes = runSims(Nreps=c(3,5,7,10,15), sim.opts=simOptions, nsims=100,
                 DEmethod="edgeR")

## for different sample sizes in two groups
simRes = runSims(Nreps=c(3,5,7,10), Nreps2=c(5,7,9,11),
                 sim.opts=simOptions, nsims=100, DEmethod="edgeR")


## End(Not run)
```

---

simRNAseq *Simulate RNA-seq count data*

Description

This function generate simulated RNA-seq count data for a two-group comparison design. It takes an object for simulation option (return object from 'RNAseq.SimOptions.2grp' function, and sample sizes (replicates in each condition, then generate a matrix of counts.

Usage

```
simRNAseq(simOptions, n1, n2)
```

Arguments

simOptions	An object for simulation option. This should be the return object from 'RNAseq.SimOptions.2grp' function.
n1, n2	Sample size in two treatment groups.

Details

The count data are generated based a negative binomial (NB) distribution. Parameters for NB are provided in the input object.

Value

A list with following fields:

`counts`	A matrix of dimension ngenes x (n1+n2). Each row is for a gene, each column is for a sample. Columns 1 to n1 are for the first condition. The rest columns are for the other condition.
`designs`	A vector/matrix representing the experimental designs. In a two-group comparison, it's a simple 0/1 vector.
`DEid`	The ID (row index) for true DE genes.
`simOptions`	The simulation option object. This is exactly the same as the input simOptions.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

RNAseq.SimOptions.2grp, runSims

Examples

```
simOptions = RNAseq.SimOptions.2grp()
data = simRNAseq(simOptions, n1=3, n2=3)
names(data)
```

---

summaryPower — *Generate a summary table for power analysis result*

Description

Takes a result object from "comparePower" function and print out a table to summarize important power-related quantities. The results are marginalized, meaning that they are averaged quantities over all strata and simulations. This provides a quick view of the marginal results for power-related quantities under all sample sizes.

Usage

```
summaryPower(powerOutput)
```

Arguments

`powerOutput`	The result object from "comparePower" function.

Value

A matrix with multiple rows, each for a different sample size. Columns include sample size, specified nomial type I control value (for FDR or p-values), actual FDR, marginal power, averaged number of true and false discoveries, and false discovery costs.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

runSims, comparePower

Examples

```
## Not run:
simOptions = RNAseq.SimOptions.2grp()
simRes = runSims(Nreps=c(3,5,7), sim.opts=simOptions, nsims=5,
                 DEmethod="edgeR")
powers = comparePower(simRes)
summaryPower(powers)

## End(Not run)
```

Index

bottomly *(param)*, 6

cheung *(param)*, 6
comparePower, 2

estParam, 4

GE.human *(param)*, 6
gilad *(param)*, 6

maqc *(param)*, 6

param, 6
pbmc *(param)*, 6
plotAll *(plots)*, 8
plotFDcost *(plots)*, 8
plotFDR *(plots)*, 8
plotPower *(plots)*, 8
plotPowerAlpha *(plots)*, 8
plotPowerFD *(plots)*, 8
plotPowerHist, 7
plotPowerTD *(plots)*, 8
plots, 8
power.seqDepth, 9

RNAseq.SimOptions.2grp, 10
runSims, 12

simRNAseq, 14
summaryPower, 15