Package ‘SPsimSeq’
March 30, 2024

Title  Semi-parametric simulation tool for bulk and single-cell RNA sequencing data

Version  1.12.0

Description  SPsimSeq uses a specially designed exponential family for density estimation to constructs the distribution of gene expression levels from a given real RNA sequencing data (single-cell or bulk), and subsequently simulates a new dataset from the estimated marginal distributions using Gaussian-copulas to retain the dependence between genes. It allows simulation of multiple groups and batches with any required sample size and library size.

License  GPL-2

Encoding  UTF-8

LazyData  true

URL  https://github.com/CenterForStatistics-UGent/SPsimSeq

Imports  stats, methods, SingleCellExperiment, fitdistrplus, graphics, edgeR, Hmisc, WGCNA, limma, mvtnorm, phyloseq, utils

biocViews  GeneExpression, RNASeq, SingleCell, Sequencing, DNASeq

RoxygenNote  7.1.0

Suggests  knitr, rmarkdown, LSD, testthat, BiocStyle

VignetteBuilder  knitr

Depends  R (>= 4.0)

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

git_branch  RELEASE_3_18

git_last_commit  e248791

git_last_commit_date  2023-10-24

Repository  Bioconductor 3.18

Date/Publication  2024-03-29

Author  Alemu Takele Assefa [aut],
        Olivier Thas [ths],
        Joris Meys [cre],
        Stijn Hawinkel [aut]

Maintainer  Joris Meys <Joris.Meys@ugent.be>
### R topics documented:

- `SPsimSeq-package` ........................................... 2
- `buildXmat` .................................................. 3
- `calculateCPM` ........................................... 3
- `checkInputValidity` ..................................... 4
- `chooseCandGenes` ....................................... 4
- `configExperiment` ....................................... 6
- `constructDens` .......................................... 7
- `estLibSizeDistr` ......................................... 7
- `evaluateDensities` ..................................... 8
- `expit` ....................................................... 9
- `extractMat` ............................................... 9
- `fitLLmodel` .............................................. 10
- `fitPoisGlm` ............................................... 10
- `fracZeroLogitModel` ................................... 11
- `genCopula` ................................................ 11
- `geneParmEst` ............................................ 12
- `genLibSizes` ............................................. 12
- `matchCopula` ............................................ 13
- `obtCorMatsBatch` ....................................... 13
- `obtCount` .................................................. 14
- `parmEstDensVec` ......................................... 14
- `prepareSPsimOutputs` .................................. 15
- `samZeroID` ................................................ 15
- `scNGP.data` ............................................... 16
- `selectGenes` .............................................. 16
- `SPsimPerGene` ........................................... 17
- `SPsimSeq` .................................................. 19
- `zeroProbModel` .......................................... 23
- `zhang.data.sub` .......................................... 24

### Description

`SPsimSeq` uses a specially designed exponential family for density estimation to constructs the distribution of gene expression levels from a given real RNA sequencing data (single-cell or bulk), and subsequently, simulates a new dataset from the estimated marginal distributions using Gaussian-copulas to retain the dependence between genes. It allows simulation of multiple groups and batches with any required sample size and library size.
buildXmat

Author(s)
Alemu Takele Assefa <alemutak@hotmail.com>
Stijn Hawinkel <stijnhawinkel@hotmail.com>

References
• Alemu Takele Assefa, Jo Vandesompele, Olivier Thas. (2020). SPsimSeq: semi-parametric simulation of bulk and single cell RNA sequencing data, Bioinformatics, , btaa105, https://doi.org/10.1093/bioinformatics/btaa105

Description
An auxiliary function to quickly construct the polynomial matrix, using Horner’s rule

Usage
buildXmat(x, nc)

Arguments
x The base
nc the number of columns

Value
A matrix with increasing powers of x in the columns

calculateCPM

Calculates counts per millions of reads, possibly with log-transform

Description
Calculates counts per millions of reads, possibly with log-transform

Usage
calculateCPM(X, const.mult, prior.count)

Arguments
X raw data matrix
const.mult a constant to multiply with
prior.count prior count to be added to the zeroes
chooseCandGenes

Value

a normalized data matrix

checkInputValidity  

Description

Check for data validity

Usage

checkInputValidity(  
s.data,  
group,  
batch,  
group.config,  
batch.config,  
w,  
log.CPM.transform,  
prior.count,  
pDE,  
lib.size.params,  
llStat.thrld,  
result.format  
)

Arguments

s.data, group, batch, group.config, batch.config, w, log.CPM.transform, prior.count, pDE, lib.size.params, llStat.thrld, result.format

see ?SPsimSeq

Value

Throws errors where needed, otherwise returns invisible

chooseCandGenes  

Select candidate genes

Description

This function can be used to independently select candidate genes from a given real RNA-srq data (bulk/single) for the SPsimSeq simulation. It chooses genes with various characteristics, such as log-fold-change above a certain threshold.
chooseCandGenes

Usage

chooseCandGenes(
  cpm.data,
  group,
  lfc.thrld,
  llStat.thrld,
  t.thrld,
  w = w,
  max.frac.zeror.diff = Inf,
  pDE,
  n.genes,
  prior.count
)

Arguments

cpm.data  logCPM transformed matrix (if log.CPM.transform=FALSE, then it is the source
          gene expression data)
group  a grouping factor
lfc.thrld  a positive numeric value for the minimum absolute log-fold-change for selecting
            candidate DE genes in the source data (when group is not NULL and pDE>0)
llStat.thrld  a positive numeric value for the minimum squared test statistics from the log-
               linear model to select candidate DE genes in the source data (when group is not
               NULL and pDE>0) containing X as a covariate to select DE genes
t.thrld  a positive numeric value for the minimum absolute t-test statistic for the log-
         fold-changes of genes for selecting candidate DE genes in the source data (when
         group is not NULL and pDE>0)
w  a numeric value between 0 and 1. The number of classes to construct the prob-
     ability distribution will be round(w*n), where n is the total number of sam-
     ples/cells in a particular batch of the source data
max.frac.zeror.diff  a numeric value >=0 indicating the maximum absolute difference in the fraction
                      of zero counts between the groups for DE genes.
pDE  fraction of DE genes
n.genes  total number of genes
prior.count  a positive constant to be added to the CPM before log transformation, to avoid
             log(0). The default is 1.

Value

a list object contating a set of candidate null and non-null genes and additional results
**Description**

Configure experiment

**Usage**

configExperiment(batch.config, group.config, tot.samples, batch, group)

**Arguments**

- **batch.config**: a numerical vector for the marginal fraction of samples in each batch. The number of batches to be simulated is equal to the size of the vector. All values must sum to 1.
- **group.config**: a numerical vector for the marginal fraction of samples in each group. The number of groups to be simulated is equal to the size of the vector. All values must sum to 1.
- **tot.samples**: total number of samples to be simulated.
- **batch, group**: batch and grouping vectors

**Value**

a list object containing the number of groups and batches to be simulated, and the experiment configuration

**Examples**

```r
batch = sample(LETTERS[1:3], 20, replace = TRUE)
group = sample(1:3, 20, replace = TRUE)

#---- a design with a total of 10 samples/cells from 1 batch and 1 group
cconfigExperiment(batch.config=1, group.config=1, tot.samples=10,
batch = batch, group = group)

#---- a design with a total of 20 samples/cells from 1 group and 2 batches with
# batch 1 has 15 samples/cells and batch 2 has 5
configExperiment(batch.config = c(15/20, 5/20), group.config = 1,
tot.samples = 20, batch = batch, group = group)

#---- a design with a total of 20 samples/cells from 1 batch and 2 groups with
# group 1 has 10 samples/cells and batch 2 has 10
configExperiment(batch.config=1, group.config=c(0.5, 0.5), tot.samples=20,
batch = batch, group = group)

#---- a design with a total of 30 samples/cells from 2 groups with group 1 has 15 samples
# and group 2 has 15, and three batches with batch 1,2, and 3 have 5, 10, and 15 samples/cells,
# respectively.
```
constructDens

Description

Construct the cumulative density

Usage

constructDens(densList.ii, exprmt.design, DE.ind.ii, returnDens = FALSE)

Arguments

densList.ii the estimated density parameters
exprmt.design experiment configuration
DE.ind.ii a boolean, is the gene to be DE?
returnDens A boolean, should densities rather than cumulative densities be returned?

Value

The cumulative density

estLibSizeDistr

Description

Estimate log-normal distribution for the library sizes

Usage

estLibSizeDistr(LS, batch)

Arguments

LS observed library sizes
batch batches

Value

Estimated log-normal parameter library sizes
evaluateDensities  Evaluate the densities in the estimated SPsimSeq object

Description

Evaluate the densities in the estimated SPsimSeq object

Usage

evaluateDensities(SPobj, newData = names(SPobj$detailed.results$densList))

Arguments

SPobj  The SPsimSeq object, with details retained
newData  A character vector of gene names

Value

da list of estimated densities, breaks and midpoints, one for every gene in newData

Examples

data("zhang.data.sub")
# filter genes with sufficient expression (important step to avoid bugs)
zhang.counts <- zhang.data.sub$counts
MYCN.status <- zhang.data.sub$MYCN.status
# simulate data
sim.data.bulk <- SPsimSeq(n.sim = 1, s.data = zhang.counts,
group = MYCN.status, n.genes = 2000, batch.config = 1,
group.config = c(0.5, 0.5), tot.samples = 20,
pDE = 0.1, lfc.thrld = 0.5, result.format = "list",
return.details = TRUE)
outDens = evaluateDensities(sim.data.bulk)
select.genes <- sample(names(outDens), 4)
select.sample = sample(
  seq_along(sim.data.bulk$detailed.results$exprmt.design$sub.groups), 1)
par(mfrow=c(2, 2))
for(i in select.genes){
  plot(outDens[[i]][[select.sample]]$mids, outDens[[i]][[select.sample]]$gy, type = "l",
     xlab = "Outcome", ylab = "Density", main = paste("Gene", i))
}
expit

**Evaluate the expit function**

**Description**
Evaluate the expit function

**Usage**
expit(x)

**Arguments**

- x the argument

**Value**

the expit of the argument

---

extractMat

*A function with S4 dispatching to extract the count matrix*

**Description**
A function with S4 dispatching to extract the count matrix

**Usage**
extractMat(Y, ...)

```r
## S4 method for signature 'SingleCellExperiment'
extractMat(Y, ...)

## S4 method for signature 'matrix'
extractMat(Y, ...)

## S4 method for signature 'data.frame'
extractMat(Y, ...)

## S4 method for signature 'phyloseq'
extractMat(Y, ...)
```

**Arguments**

- Y a matrix, data frame, phyloseq object or SingleCellExperiment
- ... additional arguments, currently ignored
Description

Fit log linear model for each gene

Usage

fitLLmodel(yy, mu.hat, sig.hat, n)

Arguments

yy
mu.hat, sig.hat
n
a list object contating a result from obtCount() function for a single gene
Carrier density estimators
number of observations

Value

a list object containing the fitted log linear model and carrier density

Description

Fast fit Poisson regression

Usage

fitPoisGlm(Ny, x, degree, offset)

Arguments

Ny, x, degree, offset

Value

see glm.fit
fracZeroLogitModel

Extract data and iterate over batches to estimate zero probability models

Description

Extract data and iterate over batches to estimate zero probability models

Usage

fracZeroLogitModel(s.data, batch, cpm.data, n.mean.class, minFracZeroes)

Arguments

s.data, cpm.data

raw and transformed data

batch

the batch vector

n.mean.class

see zeroProbModel

minFracZeroes

minimum fraction of zeroes before zero-inflation is applied

Value

a list of binomial regression parameters

genCopula

Generate a copula instance

Description

Generate a copula instance

Usage

genCopula(corMats, exprmt.design)

Arguments

corMats

List of correlation matrices

exprmt.design

Number of batches, and batch vector

Value

a list of copula instances
geneParmEst Gene level param estimates for density estimation

Description
Gene level param estimates for density estimation

Usage
geneParmEst(
cpm.data.i,
batch,
group,
prior.count = prior.count,
de.ind,
model.zero.prob,
w
)

Arguments

cpm.data.i full vector of genewise observation
batch, group batch and grouping vectors
prior.count the prior count for the cpm transofrm
de.ind a boolean, is the gene to be DE?
model.zero.prob a boolean, should zero-density be modelled?
w weight

Value
list of density estimates

genLibSizes Generate library sizes from log-normal

Description
Generate library sizes from log-normal

Usage
genLibSizes(fit.ln, exprmt.design)
matchCopula

Arguments

**fit.ln** the library size model

**exprmt.design** the design

Value

The generated library sizes per batch and group

---

**matchCopula**

*Match copulas to estimated SP distribution*

---

Description

Match copulas to estimated SP distribution

Usage

`matchCopula(cumDens, exprmt.design, copSam, sel.genes.ii)`

Arguments

**cumDens** The cumulative densities evaluated

**exprmt.design** the design

**copSam** the sampled copula

**sel.genes.ii** the gene

Value

the outcome values as a vector

---

**obtCorMatsBatch**

*A function to obtain copulas or uniform random variables*

---

Description

A function to obtain copulas or uniform random variables

Usage

`obtCorMatsBatch(cpm.data, batch)`

Arguments

**cpm.data** the transformed data matrix

**batch** the batch indicators
Value
The estimated correlation matrices per batch

\begin{verbatim}
obtCount
\end{verbatim}

Calculates height and mid points of a distribution

Description
Calculates height and mid points of a distribution

Usage
\begin{verbatim}
obtCount(Y, w)
\end{verbatim}

Arguments
\begin{itemize}
\item \textbf{Y}: a vector of gene expression data for a particular gene (in log CPM)
\item \textbf{w}: a numeric value between 0 and 1 or NULL referring the number of classes to be created
\end{itemize}

Value

\begin{verbatim}
a list object containing class breaks, mid points and counts
\end{verbatim}

\begin{verbatim}
parmEstDensVec
\end{verbatim}

Density estimation on a single vector

Description
Density estimation on a single vector

Usage
\begin{verbatim}
parmEstDensVec( Y0, model.zero.prob, min.val, w, prev.min.val = 0.25, min.countnonnull = 3 )
\end{verbatim}
prepareSPsimOutputs

Arguments

Y0 the vector of observations
model.zero.prob, min.val, w  
   see geneParmEst()
prev.min.val minimum prevalence of minimum values
min.countnonnull minimum count for estimation

Value

density estimates

prepareSPsimOutputs A function to prepare outputs

Description

A function to prepare outputs

Usage

prepareSPsimOutputs(sim.dat, exprmt.design, DE.ind, result.format, LL)

Arguments

sim.dat The simulated data
exprmt.design the design
DE.ind the differential abundance indicator
result.format the desired output format
LL simulated library sizes

Value

the data in the desired format
### Description

Return ID for observations to be set to zero

### Usage

```r
samZeroID(fracZero.logit.list, logLL, gene)
```

### Arguments

- `fracZero.logit.list`: The estimated zero model
- `logLL`: the logged library sizes
- `gene`: the gene name

### Value

A boolean, should a zero be introduced or not?

---

### Description

*Neuroblastoma NGP cells single-cell RNA-seq.*

It was retrieved from [1] (GEO accession GSE119984): This dataset is generated for a cellular perturbation experiment on the C1 instrument (SMARTer protocol) [1]. This total RNA-seq dataset contains 83 NGP neuroblastoma cells, of which 31 were treated with nutlin-3 and the other 52 cells were treated with vehicle (controls).

### Usage

```r
scNGP.data
```

### Format

A SingleCellExperiment object

### Source

GEO accession GSE119984
selectGenes

References


SingleCellExperiment  counts + gene info + cell info

Examples

data("scNGP.data")
scNGP.data

selectGenes

Sample genes from candidate genes

Description

Sample genes from candidate genes

Usage

selectGenes(pDE, exprmt.design, n.genes, null.genes0, nonnull.genes0)

Arguments

pDE fraction of genes to be made DE
exprmt.design the experiment design
n.genes the total number of genes required
null.genes0, nonnull.genes0 Candidate null and non-null genes

Value

a vector of selected genes
SPsimPerGene  
A function that generates the simulated data for a single gene

Description
A function that generates the simulated data for a single gene

Usage

```r
SPsimPerGene(
  cumDens,  
  exprmt.design, 
  sel.genes.ii, 
  log.CPM.transform, 
  prior.count, 
  LL, 
  copSam, 
  model.zero.prob, 
  fracZero.logit.list, 
  const.mult
)
```

Arguments

- `cumDens` cumulative density
- `exprmt.design` the experiment design
- `sel.genes.ii` selected gene
- `log.CPM.transform` a boolean, is log-CPM transform required?
- `prior.count` the prior count
- `LL` the library sizes
- `copSam` the generated copula
- `model.zero.prob` a boolean, should the zeroes be modelled separately
- `fracZero.logit.list` The zero model
- `const.mult` a large constant for the CPM transform, normally 1e6

Value
Simulated cpm values
SPsimSeq

A function to simulate bulk or single cell RNA sequencing data

Description

This function simulates (bulk/single cell) RNA-seq dataset from semi-parametrically estimated distributions of gene expression levels in a given real source RNA-seq dataset

Usage

SPsimSeq(
  n.sim = 1,
  s.data,
  batch = rep(1, ncol(s.data)),
  group = rep(1, ncol(s.data)),
  n.genes = 1000,
  batch.config = 1,
  group.config = 1,
  pDE = 0.1,
  cand.DE.genes = NULL,
  lfc.thrld = 0.5,
  t.thrld = 2.5,
  llStat.thrld = 5,
  tot.samples = ncol(s.data),
  model.zero.prob = FALSE,
  genewiseCor = TRUE,
  log.CPM.transform = TRUE,
  lib.size.params = NULL,
  variable.lib.size = FALSE,
  w = NULL,
  result.format = "SCE",
  return.details = FALSE,
  verbose = TRUE,
  prior.count = 1,
  const.mult = 1e+06,
  n.mean.class = 0.2,
  minFracZeroes = 0.25
)

Arguments

n.sim  an integer for the number of simulations to be generated
s.data a real source dataset (a SingleCellExperiment class or a matrix/data.frame of counts with genes in rows and samples in columns)
batch  NULL or a vector containing batch indicators for each sample/cell in the source data
group
NULL or a vector containing group indicators for each sample/cell in the source data.

n.genes
a numeric value for the total number of genes to be simulated.

batch.config
a numerical vector containing fractions for the composition of samples/cells per batch. The fractions must sum to 1. The number of batches to be simulated is equal to the size of the vector. (Example, batch.config=c(0.6, 0.4) means simulate 2 batches with 60% of the simulated samples/cells in batch 1 and the rest 40% in the second batch. Another example, batch.config=c(0.3, 0.35, 0.25) means simulate 3 batches with the first, second and third batches contain 30%, 35% and 25% of the samples/cells, respectively).

group.config
a numerical vector containing fractions for the composition of samples/cells per group. Similar definition to ‘batch.config’. The number of groups to be simulated is equal to the size of the vector. The fractions must sum to 1.

pDE
a numeric value between 0 and 1 indicating the desired fraction of DE genes in the simulated data.

cand.DE.genes
a list object containing candidate null and non-null (DE/predictor) genes. If NULL (the default), an internal function determines candidate genes based on log-fold-change and other statistics. The user can also pass a list of candidate null and non-null genes (they must be disjoint). In particular, the list should contain two character vectors (for the name of the features/genes in the source data) with names ‘null.genes’ and ‘nonnull.genes’. For example, cand.DE.genes=list(null.genes=c('A', 'B'), nonnull.genes=c('C', 'D')).

lfc.thrld
a positive numeric value for the minimum absolute log-fold-change for selecting candidate DE genes in the source data (when group is not NULL, pDE>0 and cand.DE.genes is NULL).

t.thrld
a positive numeric value for the minimum absolute t-test statistic for the log-fold-changes of genes for selecting candidate DE genes in the source data (when group is not NULL, pDE>0 and cand.DE.genes is NULL).

llStat.thrld
a positive numeric value for the minimum squared test statistics from the log-linear model to select candidate DE genes in the source data (when group is not NULL, pDE>0 and cand.DE.genes is NULL).

tot.samples
a numerical value for the total number of samples/cells to be simulated.

model.zero.prob
a logical value whether to model the zero expression probability separately (suitable for simulating of single-cell RNA-seq data or zero-inflated data).

genewiseCor
a logical value, if TRUE (default) the simulation will retain the gene-to-gene correlation structure of the source data using Gaussian-copulas. Note that if it is TRUE, the program will be slow or it may fail for a limited memory size.

log.CPM.transform
a logical value. If TRUE, the source data will be transformed into log-(CPM+const) before estimating the probability distributions.

lib.size.params
NULL or a named numerical vector containing parameters for simulating library sizes from log-normal distribution. If lib.size.params=NULL (default), then the package will fit a log-normal distribution for the library sizes in the source data.
to simulate new library sizes. If the user would like to specify the parameters
of the log-normal distribution for the desired library sizes, then the log-mean
and log-SD params of rlnorm() functions can be passed using this argument.
Example, lib.size.params = c(meanlog=10, sdog=0.2). See also ?rlnorm.

variable.lib.size
a logical value. If FALSE (default), then the expected library sizes are simulated
once and remains the same for every replication (if n.sim>1).

w
see ?hist

result.format
a character value for the type of format for the output. Choice can be 'SCE' for
SingleCellExperiment class or "list" for a list object that contains the simulated
count, column information and row information.

return.details
a logical value. If TRUE, detailed results including estimated parameters and
densities will be returned

verbose
a logical value, if TRUE a message about the status of the simulation will be
printed on the console

prior.count
a positive constant to be added to the CPM before log transformation, to avoid
log(0). The default is 1.

const.mult
A constant by which the count are scaled. Usually 1e6 to get CPM

n.mean.class
a fraction of the number of genes for the number of groups to be created for the
mean log CPM of genes

minFracZeroes
minimum fraction of zeroes before a zero inflation model is fitted

Details
This function uses a specially designed exponential family for density estimation to constructs the
distribution of gene expression levels from a given real gene expression data (e.g. single-cell or
bulk sequencing data), and subsequently, simulates a new from the estimated distributions.#' For
simulation of single-cell RNA-seq data (or any zero inflated gene expression data), the programm
involves an additional step to explicitly account for the high abundance of zero counts (if required).
This step models the probability of zero counts as a function the mean expression of the gene and
the library size of the cell (both in log scale) to add excess zeros. This can be done by using
model.zero.prob=TRUE. Note that, for extremely large size data, it is recommended to use a random
sample of cells to reduce computation time. To enable this, add the argument subset.data=TRUE
and you can specify the number of cells to be used using n.samples argument. For example
n.samples=400. Given known groups of samples/cells in the source data, DGE is simulated by
independently sampling data from distributions constructed for each group seprately. In particular,
this procedure is applied on a set of genes with absolute log-fold-change in the source data more
than a given threshold (lfc.thrld). Moreover, when the source dataset involves samples/cells pro-
cessed in different batches, our simulation procedure incorporates this batch effect in the simulated
data, if required. Different experimental designs can be simulated using the group and batch con-
figuration arguments to simulate biologica/experimental conditions and batches, respectively. Also,
it is important to filter the source data so that genes with sufficient expression will be used to estimate
the probability distributions.

Value
a list of SingleCellExperiment/list objects each containing simulated counts (not normalized), sam-
ple/cell level information in colData, and gene/feature level information in rowData.
References


Examples

```r
# Example 1: simulating bulk RNA-seq

data("zhang.data.sub")

zhang.counts <- zhang.data.sub$counts
MYCN.status <- zhang.data.sub$MYCN.status

# We simulate only a single data (n.sim = 1) with the following property
# - 1000 genes (n.genes = 1000)
# - 40 samples (tot.samples = 40)
# - the samples are equally divided into 2 groups each with 90 samples
# (group.config = c(0.5, 0.5))
# - all samples are from a single batch (batch = NULL, batch.config = 1)
# - we add 10% DE genes (pDE = 0.1)
# - the DE genes have a log-fold-change of at least 0.5 in
# the source data (lfc.thrld = 0.5)
# - we do not model the zeroes separately, they are the part of density
# estimation (model.zero.prob = FALSE)

# simulate data
set.seed(6452)
sim.data.bulk <- SPsimSeq(n.sim = 1, s.data = zhang.counts,
                          group = MYCN.status, n.genes = 1000, batch.config = 1,
                          group.config = c(0.5, 0.5), tot.samples = 40,
                          pDE = 0.1, lfc.thrld = 0.5, result.format = "list")

head(sim.data.bulk$counts[[1]][, seq_len(5)])  # count data
head(sim.data.bulk$colData)  # sample info
head(sim.data.bulk$rowData)  # gene info

# Example 2: simulating single cell RNA-seq from a single batch (read-counts)
# we simulate only a single scRNA-seq data (n.sim = 1) with the following property
# - 2000 genes (n.genes = 2000)
# - 100 cells (tot.samples = 100)
# - the cells are equally divided into 2 groups each with 50 cells
# (group.config = c(0.5, 0.5))
# - all cells are from a single batch (batch = NULL, batch.config = 1)
# - we add 10% DE genes (pDE = 0.1)
# - the DE genes have a log-fold-change of at least 0.5
# - we model the zeroes separately (model.zero.prob = TRUE)
# - the output will be in SingleCellExperiment class object (result.format = "SCE")
```

SPsimSeq
library(SingleCellExperiment)

# load the NGP nutlin data (availabl with the package, processed with
# SMARTer/C1 protocol, and contains read-counts)
data("scNGP.data")

# filter genes with sufficient expression (important step to avoid bugs)
treatment <- ifelse(scNGP.data$characteristics..treatment=="nutlin",2,1)
set.seed(654321)

# simulate data (we simulate here only a single data, n.sim = 1)
sim.data.sc <- SPsimSeq(n.sim = 1, s.data = scNGP.data, group = treatment,
  n.genes = 2000, batch.config = 1, group.config = c(0.5, 0.5),
  tot.samples = 100, pDE = 0.1, lfc.thrld = 0.5, model.zero.prob = TRUE,
  result.format = "SCE")
sim.data.sc1 <- sim.data.sc[[1]]
class(sim.data.sc1)
head(counts(sim.data.sc1)[, seq_len(5)])
colData(sim.data.sc1)
rowData(sim.data.sc1)

data.frame(zeroProbModel)

zeroProbModel  Predict zero probability using logistic regression

Description

Predict zero probability using logistic regression

Usage

zeroProbModel(cpm.data, logL, zeroMat, n.mean.class)

Arguments

cpm.data  log CPM matrix
logL      log library size of the source data
zeroMat   the matrix of zero indicators
n.mean.class  a fraction of the number of genes for the number of groups to be created for the mean log CPM of genes

Value

The coefficients of the estimated logistic regression

**Description**

The data contains 498 neuroblastoma tumors. In short, unstranded poly(A)+ RNA sequencing was performed on the HiSeq 2000 instrument (Illumina). Paired-end reads with a length of 100 nucleotides were obtained. To quantify the full transcriptome, raw fastq files were processed with Kallisto v0.42.4 (index build with GRCh38-Ensembl v85). The pseudo-alignment tool Kallisto was chosen above other quantification methods as it is performing equally good but faster. For this study, a subset of 172 tumors (samples) with high-risk disease were selected, forming two groups: the MYCN amplified ($n_1$ = 91) and MYCN non-amplified ($n_2$ = 81) tumours. Sometimes we refer this dataset to us the Zhang data or the Zhang neuroblastoma data. In this package, a subset of 5000 genes (randomly selected) are made available for illustration purpose only.

**Usage**

```r
data(zhang.data.sub)
```

**Format**

A list object

**Source**

GEOaccessionGSE49711

**References**


**Counts**

gene counts

**Group**

MYCN (0 for MYCN non-amplified and 1 for MYCN amplified)

**Examples**

```r
data("zhang.data.sub")
str(zhang.data.sub)
```
Index

* datasets
  zhang.data.sub, 24
* dataset
  scNGP.data, 16
buildXmat, 3
calculateCPM, 3
checkInputValidity, 4
chooseCandGenes, 4
cfgExperiment, 6
constructDens, 7
estLibSizeDistr, 7
evaluateDensities, 8
expit, 9
extractMat, 9
extractMat, data.frame-method
  (extractMat), 9
extractMat, matrix-method
  (extractMat), 9
extractMat, phyloseq-method
  (extractMat), 9
extractMat, SingleCellExperiment-method
  (extractMat), 9
fitLLmodel, 10
fitPoisGlm, 10
fracZeroLogitModel, 11
genCopula, 11
geneParmEst, 12
genLibSizes, 12
matchCopula, 13
obtCorMatsBatch, 13
obtCount, 14
parmEstDensVec, 14
prepareSPsimOutputs, 15
samZeroID, 16
scNGP.data, 16
selectGenes, 17
SPsimPerGene, 18
SPsimSeq, 19
SPsimSeq-package, 2
zeroProbModel, 23
zhang.data.sub, 24