# Package 'FLASHMM'

May 7, 2026

**Title** Fast and Scalable Single Cell Differential Expression Analysis using Mixed-Effects Models

**Version** 1.3.0

**Description** A fast and scalable linear mixed-effects model (LMM) estimation algorithm for analysis of single-cell differential expression. The algorithm uses summary-level statistics and requires less computer memory to fit the LMM.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** stats, MASS, Matrix

**Suggests** knitr, rmarkdown, devtools

**VignetteBuilder** knitr

**URL** https://github.com/BaderLab/FLASHMM

**BugReports** https://github.com/BaderLab/FLASHMM/issues

**NeedsCompilation** no

**Author** Changjiang Xu [aut, cre],
Delaram Pouyabahar [ctb],
Veronique Voisin [ctb],
Gary Bader [ctb]

**Maintainer** Changjiang Xu <changjiang.xu@utoronto.ca>

**Repository** CRAN

**Date/Publication** 2026-04-07 17:40:02 UTC

## Contents

---

`contrast.matrix` *Construct Contrast Matrix*

---

## Description

Construct the contrast matrix to make various comparsions of different treatments.

## Usage

```
contrast.matrix(contrast, model.matrix.names)
```

## Arguments

| | |
|---|---|
| `contrast` | A vector of character strings specifying the various comparisons, which are the expressions constituted by model.matrix.names. |
| `model.matrix.names` | Column names of model (design) matrix. |

## Value

Matrix which columns correspond to contrasts.

## Examples

```
model_variables <- c("A", "B", "C", "D")
contrast <- c("AvsB" = "A-B", "AvsC" = "A-C", 'AvsB.C.D'= "A-(B+C+D)/3")
contrast.matrix(contrast, model_variables)
```

---

`flashmm` *Fitting Linear Mixed-effects Models*

---

## Description

flashmm, a wrapper function of lmm, fits linear mixed-effects models (LMM) by sample-level data. The LMM parameters are estimated by either restricted maximum likelihood (REML) or maximum likelihood (ML) method with Fisher scoring (FS) gradient descent algorithm.

## Usage

```
flashmm(
  Y,
  fixed,
  random,
  metadata,
  is.counts = FALSE,
  method = c("REML", "ML"),
  nBlocks = NULL,
  theta0 = NULL,
  max.iter = 50,
  epsilon = 1e-05,
  output.cov = TRUE,
  output.FIM = FALSE,
  output.RE = FALSE,
  output.SS = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| Y | a features-by-samples matrix of responses. For single-cell RNA sequencing (scRNA-seq) data, Y is a genes-by-cells gene expression matrix, whose entries can be raw counts, log-transformed values, or normalized values. If Y is a matrix of raw counts, it will be log-transformed as log2(1 + Y) for differential expression analysis. |
| fixed | a one-sided model formula used to create the fixed-effects design matrix by model.matrix function. |
| random | a one-sided formula specifying a model with a single random-effects component or a list of one-sided formulas specifying multiple random-effects components, used to create the random-effects design matrix. |
| metadata | a data frame containing the variables named in fixed and random. |
| is.counts | a logical scalar indicating whether Y is a matrix of raw counts or processed values (e.g., log-transformed or normalized). If Y contains raw counts, it will be log-transformed as log2(1 + Y). |
| method | a character string, either "REML" or "ML". Defaults to "REML". If 'REML', the model is fit using REML; otherwise, using ML. |
| nBlocks | the number of blocks into which large datasets are subdivided to reduce memory usage when computing summary statistics. The default value, nBlocks = ceiling((ncol(Y) * 1e-08) * nrow(Y)), may not be adequate. If encountering the error: vector memory limit reached, you should increase the nBlocks value to avoid the issue. |
| theta0 | a vector of initial values of the variance components, (s1, ...,sk, s_(k+1)), si = sigma_i^2, the i-th variance component. s_(k+1) = sigma^2, the variance component of model residual error. |
| max.iter | the maximal number of iterations for the iterative algorithm. |

`epsilon` positive convergence tolerance. If the absolute value of the first partial derivative of log likelihood is less than epsilon, the iterations converge.

`output.cov` a logical scalar. If TRUE, output an array of the covariance matrices for the estimated coefficients, cov.beta.

`output.FIM` a logical scalar. If TRUE, output an array of the Fisher information matrices (FIM) for fixed effects, FIM.beta[,,i] = $X^T(Cov(Y[i,])^{-1}X$ for the i-th feature, and variance components, FIM.theta[,,i] = $Var(\partial(logL_i)/\partial(theta)) = -E[\partial^2(logL_i)/\partial(theta)\partial(theta^T)]$ for the i-th feature.

`output.RE` a logical scalar. If TRUE, output the best linear unbiased prediction (BLUP) of the random effects.

`output.SS` a logical scalar. If TURE, output list(XX, XY, ZX, ZY, ZZ, Ynorm, n), a list of the summary-level data (summary statistics), defined by XX=t(X)%*%X, XY=t(Y%*%X), ZX=t(Z)%*%X, ZY=t(Z)%*%Y, ZZ=t(Z)%*%Z, Ynorm=rowSums(Y*Y), and n=nrow(X), which can also be computed using the sslmm function.

`verbose` a logical scalar. If TRUE, print the number of features for which LMM fitting did not converge (abs(dlogL) > epsilon).

## Value

A list containing the following components:

`method` the method, either REML or ML, for estimating the LMM parameters (fixed effects and variance components).

`epsilon` convergence tolerance.

`dlogL` first partial derivatives of log-likelihoods for each feature.

`logLik` maximum log-likelihoods for ML method or maximum log-restricted-likelihood for REML method.

`niter` numbers of iterations for each feature.

`coef` a matrix of estimated coefficients (fixed effects or beta), each column corresponds to a feature and each row one covariate.

`se` a matrix of standard errors of the estimated coefficients.

`t` a matrix of t-values for the fixed effects, equal to coef/se.

`p` a matrix of two-sided p-values for the t-tests of the fixed effects.

`df` degrees of freedom for the t-statistics (values).

`cov` a array of covariance matrices of the estimated coefficients (fixed effects).

`theta` a matrix of the estimated variance components, each column corresponds to a feature and each row one variance component. The last row is the variance component of the residual error.

`se.theta` standard errors of the estimated theta.

`FIM.beta` the Fisher information matrices (FIM) for fixed effects.

`FIM.theta` the Fisher information matrices (FIM) for variance components.

`RE` a matrix of the best linear unbiased prediction (BLUP) of random effects.

`summary_data` a list of the summary-level data (summary statistics).

## See Also

lmmfit, lmm

## Examples

```
#Generate data
set.seed(2024)

#Y: gene expression matrix
n <- 1e3
m <- 10
Y <- matrix(rnorm(m*n), m, n)
rownames(Y) <- paste0("Gene", 1:nrow(Y))

#metadata: samples (sam) and treatments (trt)
metadata <- data.frame(trt = character(n), sam = character(n))
metadata$trt <- sample(c("A", "B"), n, replace = TRUE)
q <- 10
metadata$sam <- paste0("S", sample.int(q, n, replace = TRUE))

#Model formulas
fixed <- ~ 0 + trt
random <- ~ 0 + sam

#Fit LMM by flashmm with the model formulas based on the cell-level data
fit1 <- flashmm(Y, fixed, random, metadata = metadata, is.counts = FALSE)

#Fit LMM by lmmfit with the model design matrices based on the cell-level data
#design matrices
X <- model.matrix(fixed, metadata)
Z <- model.matrix(random, metadata)
d <- ncol(Z)
fit2 <- lmmfit(Y, X, Z, d = d)

identical(fit1, fit2)

#Fit LMM by lmm based on summary-level data
#Compute and store the summary-level data:
n <- nrow(X)
XX <- t(X)%*%X
XY <- t(Y%*%X)
ZX <- t(Z)%*%X
ZY <- t(Y%*%Z)
ZZ <- t(Z)%*%Z
Ynorm <- rowSums(Y*Y)
fit3 <- lmm(XX, XY, ZX, ZY, ZZ, Ynorm = Ynorm, n = n, d = d)

identical(fit2, fit3)

#Hypothesis testing
lmmtest(fit1)
lmmtest(fit1, index = 2)
```

```
lmmtest(fit1, contrast = cbind("B-A" = c(-1, 1)))
```

---

`lmm` *Fitting Linear Mixed-effects Models*

---

## Description

lmm is used to fit a linear mixed-effects model (LMM) based on summary-level data. The LMM parameters are estimated by either restricted maximum likelihood (REML) or maximum likelihood (ML) method with Fisher scoring (FS) gradient descent algorithm.

## Usage

```
lmm(
  XX,
  XY,
  ZX,
  ZY,
  ZZ,
  Ynorm,
  n,
  d = ncol(ZZ),
  method = c("REML", "ML"),
  theta0 = NULL,
  max.iter = 50,
  epsilon = 1e-05,
  output.cov = TRUE,
  output.FIM = FALSE,
  output.RE = FALSE,
  output.SS = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `XX` | = t(X)%*%X, where X is a design matrix for fixed effects. |
| `XY` | = t(Y%*%X), where Y is a features-by-samples matrix of observed responses (genes-by-cells expression matrix for scRNA-seq). |
| `ZX` | = t(Z)%*%X, where Z = [Z1, ..., Zk] is a design matrix for k random components (factors). |
| `ZY` | = t(Y%*%Z). |
| `ZZ` | = t(Z)%*%Z. |
| `Ynorm` | = rowSums(Y*Y), norms for features (each row). |
| `n` | = nrow(X), number of samples (cells in scRNA-seq). |

| | |
|---|---|
| d | = (d1,...,dk), where di = ncol(Zi), the number of columns in Zi. sum(d) = ncol(Z), the number of columns in Z. For the model with only one random component, d = ncol(Z). |
| method | a character string, either "REML" or "ML". Defaults to "REML". If 'REML', the model is fit using REML; otherwise, using ML. |
| theta0 | a vector of initial values of the variance components, (s1, ...,sk, s_(k+1)), si = sigma_i^2, the i-th variance component. s_(k+1) = sigma^2, the variance component of model residual error. |
| max.iter | the maximal number of iterations for the iterative algorithm. |
| epsilon | positive convergence tolerance. If the absolute value of the first partial derivative of log likelihood is less than epsilon, the iterations converge. |
| output.cov | a logical scalar. If TRUE, output an array of the covariance matrices for the estimated coefficients, cov.beta. |
| output.FIM | a logical scalar. If TRUE, output an array of the Fisher information matrices (FIM) for fixed effects, FIM.beta[,,i] = $X^T(Cov(Y[i,])^{-1}X$ for the i-th feature, and variance components, FIM.theta[,,i] = $Var(\partial(logL_i)/\partial(theta)) = -E[\partial^2(logL_i)/\partial(theta)\partial(theta^T)]$ for the i-th feature. |
| output.RE | a logical scalar. If TRUE, output the best linear unbiased prediction (BLUP) of the random effects. |
| output.SS | a logical scalar. If TURE, output list(XX, XY, ZX, ZY, ZZ, Ynorm, n), a list of the summary-level data (summary statistics), which can also be computed using the sslmm function. |
| verbose | a logical scalar. If TRUE, print the number of features for which LMM fitting did not converge (abs(dlogL) > epsilon). |

## Value

A list containing the following components:

| | |
|---|---|
| method | the method, either REML or ML, for estimating the LMM parameters (fixed effects and variance components). |
| epsilon | convergence tolerance. |
| dlogL | first partial derivatives of log-likelihoods for each feature. |
| logLik | maximum log-likelihoods for ML method or maximum log-restricted-likelihood for REML method. |
| niter | numbers of iterations for each feature. |
| coef | a matrix of estimated coefficients (fixed effects or beta), each column corresponds to a feature and each row one covariate. |
| se | a matrix of standard errors of the estimated coefficients. |
| t | a matrix of t-values for the fixed effects, equal to coef/se. |
| p | a matrix of two-sided p-values for the t-tests of the fixed effects. |
| df | degrees of freedom for the t-statistics (values). |
| cov | a array of covariance matrices of the estimated coefficients (fixed effects). |

| | |
|---|---|
| theta | a matrix of the estimated variance components, each column corresponds to a feature and each row one variance component. The last row is the variance component of the residual error. |
| se.theta | standard errors of the estimated theta. |
| FIM.beta | the Fisher information matrices (FIM) for fixed effects. |
| FIM.theta | the Fisher information matrices (FIM) for variance components. |
| RE | a matrix of the best linear unbiased prediction (BLUP) of random effects. |
| summary_data | a list of the summary-level data (summary statistics). |

## Examples

```
#Generate data: X, Y, and Z.
set.seed(2024)

n <- 1e3
m <- 10
Y <- matrix(rnorm(m*n), m, n)
rownames(Y) <- paste0("Gene", 1:nrow(Y))

trt <- sample(c("A", "B"), n, replace = TRUE)
X <- model.matrix(~ 0 + trt)

q <- 20
sam <- rep(NA, n)
sam[trt == "A"] <- paste0("A", sample.int(q/2, sum(trt == "A"), replace = TRUE))
sam[trt == "B"] <- paste0("B", sample.int(q/2, sum(trt == "B"), replace = TRUE))
Z <- model.matrix(~ 0 + sam)
d <- ncol(Z)

#Fit LMM by summary-level data
#Compute and store the summary-level data:
n <- nrow(X)
XX <- t(X)%*%X
XY <- t(Y%*%X)
ZX <- t(Z)%*%X
ZY <- t(Y%*%Z)
ZZ <- t(Z)%*%Z
Ynorm <- rowSums(Y*Y)
fit <- lmm(XX, XY, ZX, ZY, ZZ, Ynorm = Ynorm, n = n, d = d)
str(fit)
```

---

# lmmfit *Fitting Linear Mixed-effects Models*

---

## Description

lmmfit, a wrapper function of lmm, fits linear mixed-effects models (LMM) by sample-level data. The LMM parameters are estimated by either restricted maximum likelihood (REML) or maximum likelihood (ML) method with Fisher scoring (FS) gradient descent algorithm.

## Usage

```
lmmfit(
  Y,
  X,
  Z,
  d = ncol(Z),
  method = c("REML", "ML"),
  nBlocks = ceiling((ncol(Y) * 1e-08) * nrow(Y)),
  theta0 = NULL,
  max.iter = 50,
  epsilon = 1e-05,
  output.cov = TRUE,
  output.FIM = FALSE,
  output.RE = FALSE,
  output.SS = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| Y | a features-by-samples matrix of responses. For single-cell RNA sequencing (scRNA-seq) data, Y is a genes-by-cells gene expression matrix, whose entries can be raw counts, log-transformed values, or normalized values. If Y is a matrix of raw counts, it will be log-transformed as log2(1 + Y) for differential expression analysis. |
| X | a design matrix for fixed effects, with rows corresponding to the columns of Y. |
| Z | [Z1, ..., Zk] is a design matrix for k random components (factors), with rows corresponding to the columns of Y. |
| d | (d1,...,dk), where di = ncol(Zi), the number of columns in Zi. sum(d) = ncol(Z), the number of columns in Z. For the model with only one random component, d = ncol(Z). |
| method | a character string, either "REML" or "ML". Defaults to "REML". If 'REML', the model is fit using REML; otherwise, using ML. |
| nBlocks | the number of blocks into which large datasets are subdivided to reduce memory usage when computing summary statistics. The default value, nBlocks = ceiling((ncol(Y)*1e-08)*nrow(Y)), may not be adequate. If encountering the error: vector memory limit reached, you should increase the nBlocks value to avoid the issue. |
| theta0 | a vector of initial values of the variance components, (s1, ...,sk, s_(k+1)), si = sigma_i^2, the i-th variance component. s_(k+1) = sigma^2, the variance component of model residual error. |
| max.iter | the maximal number of iterations for the iterative algorithm. |
| epsilon | positive convergence tolerance. If the absolute value of the first partial derivative of log likelihood is less than epsilon, the iterations converge. |
| output.cov | a logical scalar. If TRUE, output an array of the covariance matrices for the estimated coefficients, cov.beta. |

`output.FIM` a logical scalar. If TRUE, output an array of the Fisher information matrices (FIM) for fixed effects, FIM.beta[,,i] = $X^T(Cov(Y[i,])^{-1}X$ for the i-th feature, and variance components, FIM.theta[,,i] = $Var(\partial(logL_i)/\partial(theta)) = -E[\partial^2(logL_i)/\partial(theta)\partial(theta^T)]$ for the i-th feature.

`output.RE` a logical scalar. If TRUE, output the best linear unbiased prediction (BLUP) of the random effects.

`output.SS` a logical scalar. If TURE, output list(XX, XY, ZX, ZY, ZZ, Ynorm, n), a list of the summary-level data (summary statistics), defined by XX=t(X)%*%X, XY=t(Y%*%X), ZX=t(Z)%*%X, ZY=t(Z)%*%Y, ZZ=t(Z)%*%Z, Ynorm=rowSums(Y*Y), and n=nrow(X), which can also be computed using the sslmm function.

`verbose` a logical scalar. If TRUE, print the number of features for which LMM fitting did not converge (abs(dlogL) > epsilon).

## Value

A list containing the following components:

`method` the method, either REML or ML, for estimating the LMM parameters (fixed effects and variance components).

`epsilon` convergence tolerance.

`dlogL` first partial derivatives of log-likelihoods for each feature.

`logLik` maximum log-likelihoods for ML method or maximum log-restricted-likelihood for REML method.

`niter` numbers of iterations for each feature.

`coef` a matrix of estimated coefficients (fixed effects or beta), each column corresponds to a feature and each row one covariate.

`se` a matrix of standard errors of the estimated coefficients.

`t` a matrix of t-values for the fixed effects, equal to coef/se.

`p` a matrix of two-sided p-values for the t-tests of the fixed effects.

`df` degrees of freedom for the t-statistics (values).

`cov` a array of covariance matrices of the estimated coefficients (fixed effects).

`theta` a matrix of the estimated variance components, each column corresponds to a feature and each row one variance component. The last row is the variance component of the residual error.

`se.theta` standard errors of the estimated theta.

`FIM.beta` the Fisher information matrices (FIM) for fixed effects.

`FIM.theta` the Fisher information matrices (FIM) for variance components.

`RE` a matrix of the best linear unbiased prediction (BLUP) of random effects.

`summary_data` a list of the summary-level data (summary statistics).

## See Also

`lmm`

## Examples

```
#Generate data: X, Y, and Z.
set.seed(2024)

n <- 1e3
m <- 10
Y <- matrix(rnorm(m*n), m, n)
rownames(Y) <- paste0("Gene", 1:nrow(Y))

trt <- sample(c("A", "B"), n, replace = TRUE)
X <- model.matrix(~ 0 + trt)

q <- 20
sam <- rep(NA, n)
sam[trt == "A"] <- paste0("A", sample.int(q/2, sum(trt == "A"), replace = TRUE))
sam[trt == "B"] <- paste0("B", sample.int(q/2, sum(trt == "B"), replace = TRUE))
Z <- model.matrix(~ 0 + sam)
d <- ncol(Z)

#Fit LMM by the cell-level data
fit <- lmmfit(Y, X, Z, d = d)
str(fit)

#Fit LMM by summary-level data
#Compute and store the summary-level data:
n <- nrow(X)
XX <- t(X)%*%X
XY <- t(Y%*%X)
ZX <- t(Z)%*%X
ZY <- t(Y%*%Z)
ZZ <- t(Z)%*%Z
Ynorm <- rowSums(Y*Y)
fitss <- lmm(XX, XY, ZX, ZY, ZZ, Ynorm = Ynorm, n = n, d = d)

identical(fit, fitss)

#Hypothesis testing
lmmtest(fit)
lmmtest(fit, index = 2)
lmmtest(fit, contrast = cbind("B-A" = c(-1, 1)))
```

---

`lmmtest` *Testing Fixed Effects and Contrasts of the Fixed Effects*

---

## Description

lmmtest is used to test fixed effects or contrasts of fixed effects by t-statistic.

## Usage

```
lmmtest(
  fit,
  index,
  contrast = NULL,
  alternative = c("two.sided", "less", "greater")
)
```

## Arguments

| | |
|---|---|
| fit | Output of lmmfit or lmm, which contains coef (estimates of fixed effects), a matrix with rows representing the fixed effects and columns the different response variables in the model, cov (covariance matrix of the fixed effects), an array of three dimensions for different response variables in the model, df (residual degree of freedom in the linear model). |
| index | A vector of integers or characters indicating which fixed effects are to be tested. By default index consists of all of the fixed effects. Ignored if contrast is not NULL. |
| contrast | A matrix with columns corresponding to contrasts of the fixed effects to be tested. |
| alternative | A character string specifying the alternative hypothesis, one of "two.sided", "greater" or "less". |

## Value

A matrix of coefficients, t-values and p-values, in which the rows correspond to the features and the columns the fixed effects (covariates). .

---

# simuRNAseq *Simulating Multi-sample Multi-cell-type scRNA-seq Dataset based on Negative Binomial Distribution*

---

## Description

simuRNAseq simulates scRNA-seq data with multiple subjects (samples), multiple clusters (cell-types) and two treatments (conditions) based on a negative binomial (NB) distribution using a reference data as background or control. The reference data consisting of genes-by-cells counts matrix is used to estimate the NB dispersion and means for the genes to be simulated.

The simulated genes are randomly selected from the reference data. If the number of simulated genes is equal to the number of genes in the reference data, the original gene names in the reference data are retained. The NB dispersion are estimated by the method-of-moments estimate (MME). The NB means for the background in the control are estimated by the sample mean. The NB means for the differentially expressed (DE) genes are given by the sample mean plus a log-fold change (logFC).

The simulated cells are randomly selected from the meta data that specifies subjects, cell-types and treatments for the cells. The meta data consists of samples, clusters of cell types, and treatments, which can be generated either from reference data or randomly. If not provided, it will be randomly generated.

A random seed is recommended to be specified by set.seed before simulating data.

## Usage

```
simuRNAseq(
  counts,
  nGenes = nrow(counts),
  nCells = ncol(counts),
  metadata = NULL,
  samples.nested = TRUE,
  nsam = 25,
  ncls = 10,
  ntrt = 2,
  trt = NULL,
  nDEgenes = ncls,
  nDEgenesType,
  pDEgenesType = NULL,
  adjust.library.size = TRUE,
  direction = c("both", "up", "down"),
  minbeta = 0.25,
  maxbeta = 1,
  var.randomeffects = 0.1
)
```

## Arguments

| | |
|---|---|
| counts | A genes-by-cells matrix of reference counts. If missing, counts is generated by a negative binomial distribution. |
| nGenes | Number of genes to be simulated. |
| nCells | Number of cells to be simulated. |
| metadata | The meta data consisting of 4 columns: sam (sample labels), cls (cluster lables of cell types), trt (treatments or conditions) and libsize (library size or total counts per cell), which is randomly generated if not provided. |
| samples.nested | If TRUE, when metadata is not provided, each simulated subject (sample) belongs to only one condition (either treatment or control), that is, the subject is nested in a condition (treatment). |
| nsam | Number of subjects (individuals). |
| ncls | Number of clusters (cell-types). |
| ntrt | Number of treatments (only one condition is treated). |
| trt | Treatment, specifying which condition is treated. |
| nDEgenes | Total number of DE genes. |
| nDEgenesType | Number of DE genes specific to a cell type, named by cell cluster labels. |

pDEgenesType Proportion of DE genes in a cell-type. Default NULL means equal proportion.

adjust.library.size
If TRUE, adjust library sizes using the reference counts.

direction Specify if the DE genes are up- and/or down-regulated.

minbeta Lower bound of the DE gene logFC.

maxbeta Upper bound of the DE gene logFC. minbeta < maxbeta. If direction = "both", minbeta*maxbeta > 0. If direction = "up", minbeta > 0. If direction = "down", maxbeta < 0.

var.randomeffects
Variance of random effects

## Value

A list containing the following components:

ref.mean.dispersion
A data frame of the reference counts' means and dispersion.

metadata Meta data consisting of 4 columns: sam (sample labels), cls (cluster lables of cell types), trt (two treatments/conditions) and libsize (library sizes).

counts A genes-by-cells matrix of the simulated counts.

DEgenes A data frame of DE genes consisting of 3 columns: gene, beta (effect), and cluster to which the gene is specific.

treatment The condition treated.

## Examples

```
#Simulate a multi-sample multi-cell-type scRNA-seq dataset.
set.seed(2412)
refdata <- simuRNAseq(nGenes = 50, nCells = 1000, nsam = 25, ncls = 4, ntrt = 2, nDEgenes = 6)
str(refdata)
#The samples are nested in a condition.
table(refdata$metadata[, c("sam", "trt")])

#Simulate a multi-sample multi-cell-type scRNA-seq dataset with reference data.
dat <- simuRNAseq(refdata$counts)
str(dat)

all(rownames(dat$counts) == rownames(refdata$counts))
all(colnames(dat$counts) == colnames(refdata$counts))

#Analyze differentially expressed (DE) genes specific to a cell-type using LMM.
Y <- log(dat$counts + 1) #expressions (log-transformed counts)
X <- model.matrix(~ 0 + log(libsize) + cls + cls:trt, data = dat$metadata)
Z <- model.matrix(~ 0 + sam, data = dat$metadata)
d <- ncol(Z)

#Fit LMM using cell-level data.
fit <- lmmfit(Y, X, Z, d = d)
```

```
#Fit LMM using summary-level data.
#Compute and store the summary-level data:
n <- nrow(X)
XX <- t(X)%*%X
XY <- t(Y%*%X)
ZX <- t(Z)%*%X
ZY <- t(Y%*%Z)
ZZ <- t(Z)%*%Z
Ynorm <- rowSums(Y*Y)
fitss <- lmm(XX, XY, ZX, ZY, ZZ, Ynorm = Ynorm, n = n, d = d)

identical(fit, fitss)

#Hypothesis testing
test <- lmmtest(fit)
head(test)

#The DE genes specific to a cell-type.
tail(test[, grep(":", colnames(test))])
```

---

`sslmm` *Computing Summary-level Data from Individual-level Data*

---

## Description

sslmm can be used to compute the summary statistics (summary-level data) for `lmm` function, defined as

- XX = t(X)%*%X
- XY = t(X)%*%t(Y)
- ZX = t(Z)%*%X
- ZY = t(Z)%*%t(Y)
- ZZ = t(Z)%*%Z
- Ynorm = rowSums(Y*Y)
- n = nrow(X)

## Usage

```
sslmm(X, Y, Z, nBlocks = ceiling((ncol(Y) * 1e-08) * nrow(Y)))
```

## Arguments

| | |
|---|---|
| `X` | A design matrix for fixed effects, with rows corresponding to the columns of Y. |
| `Y` | A features-by-samples matrix of responses (genes-by-cells matrix of gene expressions for scRNA-seq). |

| | |
|---|---|
| Z | A design matrix for random effects, with rows corresponding to the columns of Y. |
| nBlocks | Number of the blocks, which a big data is subdivided into, used for reducing the storage in computing the summary statistics that are computed from a block of data. The default value may not be adequate. If encountering the error: vector memory limit reached, you should increase the nBlocks value to avoid the issue. |

## Value

A list of summary statistics: XX, XY, ZX, ZY, ZZ, Ynorm and n.

## Examples

```
n <- 1e3
set.seed(2024)
p <- 2
X <- matrix(rnorm(p*n), n, p)
colnames(X) <- paste0("X", 1:ncol(X))
m <- 3
Y <- matrix(rnorm(m*n), m, n)
rownames(Y) <- paste0("Y", 1:nrow(Y))
q <- 4
Z <- gl(q, n/q, labels = letters[1:q])
Z <- model.matrix(~ 0 + Z)
sslmm(X, Y, Z)

s1 <- sslmm(X, Y, Z, nBlocks = 1)
s2 <- sslmm(X, Y, Z, nBlocks = 2)
s3 <- sslmm(X, Y, Z, nBlocks = 3)

identical(s1, s2)
identical(s2, s3)
```

# Index