Package 'twosigma'

December 13, 2021

Type Package

Title DE Analysis for Single-Cell RNA-Sequencing Data

Version 1.0.2

Date 2021-12-10

Maintainer Eric Van Buren <edvanburen@gmail.com>

Description Implements the TWO-Component Single Cell Model-Based Association Method (TWO-SIGMA) for gene-level differential expression (DE) analysis and DE-based gene set testing of single-cell RNA-sequencing datasets. See Van Buren et al. (2020) <doi:10.1002/gepi.22361> and Van Buren et al. (2021) <doi:10.1101/2021.01.24.427979>.

License AGPL-3

Imports multcomp (>= 1.4-13), glmmTMB, methods, pscl (>= 1.5.5), pbapply (>= 1.4.0), parallel (>= 3.6.3), doParallel (>= 1.0.15)

Encoding UTF-8

LazyData false

URL https://github.com/edvanburen/twosigma

BugReports https://github.com/edvanburen/twosigma/issues

RoxygenNote 7.1.0

Suggests testthat

NeedsCompilation no

Author Eric Van Buren [aut, cre],
Yun Li [aut],
Di Wu [aut],
Ming Hu [aut]

Repository CRAN

Date/Publication 2021-12-13 09:40:02 UTC

R topics documented:

`adhoc.twosigma` *adhoc.twosigma: Perform the ad hoc method described in TWO-SIGMA paper*

Description

adhoc.twosigma: Perform the ad hoc method described in TWO-SIGMA paper

Usage

```
adhoc.twosigma(
  count,
  mean_covar,
  zi_covar,
  id,
  weights = rep(1, length(count))
)
```

Arguments

`count`	Vector of non-negative integer read counts.
`mean_covar`	Covariates for the (conditional) mean model. Must be a matrix (without an intercept column) or = 1 to indicate an intercept only model.
`zi_covar`	Covariates for the zero-inflation model. Must be a matrix (without an intercept column), = 1 to indicate an intercept only model, or = 0 to indicate no zero-inflation model desired.
`id`	Vector of individual-level ID's. Used as predictor in ANOVA model.
`weights`	weights, as in glm. Defaults to 1 for all observations and no scaling or centering of weights is performed. Passed into zeroinfl function.

Value

P-value from the ANOVA F test.

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
rownames(sim_dat)<-paste("Gene",1:2)

# Run adhoc.twosigma

adhoc.twosigma(sim_dat[1,],mean_covar = X,zi_covar=Z,id = id)
```

`lr.twosigma` *Convenient wrapper function for performing joint likelihood ratio tests using the TWO-SIGMA model.*

Description

Convenient wrapper function for performing joint likelihood ratio tests using the TWO-SIGMA model.

Usage

```
lr.twosigma(
  count_matrix,
  mean_covar,
  zi_covar,
  covar_to_test,
  mean_re = FALSE,
  zi_re = FALSE,
  id,
  return_full_fits = TRUE,
  adhoc = FALSE,
  adhoc_thresh = 0.1,
  silent = FALSE,
  disp_covar = NULL,
  weights = rep(1, ncol(count_matrix)),
  control = glmmTMBControl(),
  ncores = 1,
  cluster_type = "Fork",
  chunk_size = 10,
  lb = FALSE
)
```

Arguments

`count_matrix`	Matrix of non-negative integer read counts, with rows corresponding to genes and columns corresponding to cells. It is recommended to make the rownames the gene names for better output.
`mean_covar`	Covariates for the (conditional) mean model. Must be a matrix (without an intercept column) or a vector if a single covariate is being tested.
`zi_covar`	Covariates for the zero-inflation model. Must be a matrix (without an intercept column) or a vector if a single covariate is being tested.
`covar_to_test`	Either a string indicating the column name of the covariate to test or an integer referring to its column position in BOTH the mean_covar and zi_covar matrices (if the two matrices differ using a string name is preferred). Argument is ignored if mean_covar and zi_covar are both a single covariate (that covariate is assumed of interest).
`mean_re`	Should random intercepts be included in the (conditional) mean model?
`zi_re`	Should random intercepts be included in the zero-inflation model?
`id`	Vector of individual-level ID's. Used for random effect prediction and the adhoc method but required regardless.
`return_full_fits`	If TRUE, fit objects of class glmmTMB are returned. If FALSE, only objects of class summary.glmmTMB are returned. The latter require a much larger amount of memory to store.
`adhoc`	Should the adhoc method be used by default to judge if random effects are needed?
`adhoc_thresh`	Value below which the adhoc p-value is deemed significant (and thus RE are deemed necessary). Only used if adhoc==TRUE.

`silent`	If TRUE, progress is not printed.
`disp_covar`	Covariates for a log-linear model for the dispersion. Either a matrix or = 1 to indicate an intercept only model.
`weights`	weights, as in glm. Defaults to 1 for all observations and no scaling or centering of weights is performed. See `?glmmTMBControl`.
`control`	Control parameters for optimization in glmmTMB.
`ncores`	Number of cores used for parallelization. Defaults to 1, meaning no parallelization of any kind is done.
`cluster_type`	Whether to use a "cluster of type "Fork" or "Sock". On Unix systems, "Fork" will likely improve performance. On Windows, only "Sock" will actually result in parallelized computing.
`chunk_size`	Number of genes to be sent to each parallel environment. Parallelization is more efficient, particularly with a large count matrix, when the count matrix is 'chunked' into some common size (e.g. 10, 50, 200). Defaults to 10.
`lb`	Should load balancing be used for parallelization? Users will likely want to set to FALSE for improved performance.

Value

A list with the following elements:

- `fit_null:` Model fits under the null hypothesis. If `return_summary_fits=TRUE`, returns a list of objects of class `summary.glmmTMB`. If `return_summary_fits=FALSE`, returns a list of model fit objects of class `glmmTMB`. In either case, the order matches the row order of `count_matrix`, and the names of the list elements are taken as the rownames of `count_matrix`.
- `fit_alt:` Model fits under the alt hypothesis of the same format as `fit_null`.
- `LR_stat:` Vector of Likelihood Ratio statistics. A value of 'NA' implies a convergence issue or other model fit problem.
- `LR_p.val:` Vector of Likelihood Ratio p-values. A value of 'NA' implies a convergence issue or other model fit problem.
- `adhoc_include_RE:` Logical vector indicator whether the adhoc method determined random effects needed. If `adhoc=F`, then a vector of NA's.

Details

This function assumes that the variable being tested is in both components of the model (and thus that the zero-inflation component exists and contains more than an Intercept). Users wishing to do fixed effect testing in other cases or specify custom model formulas they will need to construct the statistics themselves using either two separate calls to `twosigma` or the `lr.twosigma_custom` function. If `adhoc=TRUE`, any input in mean_re and zi_re will be ignored. If either model fails to converge, or the LR statistic is negative, both the statistic and p-value are assigned as NA.

Examples

```
# Set Parameters to Simulate Some Data
```

```
nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
rownames(sim_dat)<-paste("Gene",1:2)

# Run lr.twosigma

lr.twosigma(count=sim_dat[1,,drop=FALSE],mean_covar = X,zi_covar = Z,id=id,covar_to_test = 1)
```

`lr.twosigma_custom` *Convenient wrapper function for performing joint likelihood ratio tests with the TWO-SIGMA model using custom user-specified formulas.*

Description

Convenient wrapper function for performing joint likelihood ratio tests with the TWO-SIGMA model using custom user-specified formulas.

Usage

```
lr.twosigma_custom(
  count_matrix,
  mean_form_alt,
  zi_form_alt,
  mean_form_null,
  zi_form_null,
  id,
  lr.df,
  return_full_fits = TRUE,
  disp_covar = NULL,
  weights = rep(1, ncol(count_matrix)),
  control = glmmTMBControl(),
  ncores = 1,
  cluster_type = "Fork",
  chunk_size = 10,
  lb = FALSE,
  internal_call = FALSE
)
```

Arguments

`count_matrix`	Matrix of non-negative integer read counts, with rows corresponding to genes and columns corresponding to cells. It is recommended to make the rownames the gene names for better output.
`mean_form_alt`	Custom two-sided model formula for the (conditional) mean model under the null. Formula is passed directly into glmmTMB with random effects specified as in the lme4 package. Users should ensure that the dependent variable matches the argument to the parameter "count."
`zi_form_alt`	Custom one-sided model formula for the zero-inflation model under the alternative. Formula is passed directly into glmmTMB with random effects specified as in lme4.
`mean_form_null`	Custom two-sided model formula for the (conditional) mean model under the null. Syntax is as in `mean_form_alt`.
`zi_form_null`	Custom one-sided model formula for the zero-inflation model under the null. Syntax is as in `zi_form_alt`.
`id`	Vector of individual-level (sample-level) ID's. Used for random effect prediction but required regardless of their presence in the model.
`lr.df`	Degrees of Freedom for the constructed likelihood ratio test. Must be a non-negative integer.
`return_full_fits`	If TRUE, full fit objects of class glmmTMB are returned. If FALSE, only fit objects of class summary.glmmTMB are returned. The latter requires far less memory to store.
`disp_covar`	Covariates for a log-linear model for the dispersion. Either a matrix or = 1 to indicate an intercept only model.
`weights`	weights, as in glm. Defaults to 1 for all observations and no scaling or centering of weights is performed.
`control`	Control parameters for optimization in glmmTMB. See `?glmmTMBControl`.
`ncores`	Number of cores used for parallelization. Defaults to 1, meaning no parallelization of any kind is done.

`cluster_type`	Whether to use a "cluster of type "Fork" or "Sock". On Unix systems, "Fork" will likely improve performance. On Windows, only "Sock" will actually result in parallelized computing.
`chunk_size`	Number of genes to be sent to each parallel environment. Parallelization is more efficient, particularly with a large count matrix, when the count matrix is 'chunked' into some common size (e.g. 10, 50, 200). Defaults to 10.
`lb`	Should load balancing be used for parallelization? Users will likely want to set to FALSE for improved performance.
`internal_call`	Not needed by users called `lr.twosigma_custom` directly.

Value

A list with the following elements:

- `fit_null`: Model fits under the null hypothesis. If `return_summary_fits=TRUE`, returns a list of objects of class `summary.glmmTMB`. If `return_summary_fits=FALSE`, returns a list of model fit objects of class `glmmTMB`. In either case, the order matches the row order of `count_matrix`, and the names of the list elements are taken as the rownames of `count_matrix`.
- `fit_alt`: Model fits under the alt hypothesis of the same format as `fit_null`.
- `LR_stat`: Vector of Likelihood Ratio statistics. A value of 'NA' implies a convergence issue or other model fit problem.
- `LR_p.val`: Vector of Likelihood Ratio p-values. A value of 'NA' implies a convergence issue or other model fit problem.

Details

This function is a wrapper for conducting fixed effect likelihood ratio tests with twosigma. There is no checking to make sure that the alt and null model formulas represent a valid likelihood ratio test when fit together. Users must ensure that inputted formulas represent valid nested models. If either model fails to converge, or the LR statistic is negative, both the statistic and p-value are assigned as NA.

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)
```

```
# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
rownames(sim_dat)<-paste("Gene",1:2)

# Run lr.twosigma_custom

lr.twosigma_custom(count=sim_dat[1,,drop=FALSE]
,mean_form_alt = count~X,mean_form_null = count~X[,-1]
,zi_form_alt = ~0,zi_form_null = ~0,id=id,lr.df=1)
```

---

simulate_zero_inflated_nb_random_effect_data
Simulated zero-inflated negative binomial data with random effects

---

Description

Simulated zero-inflated negative binomial data with random effects

Usage

```
simulate_zero_inflated_nb_random_effect_data(
  ncellsper,
  X,
  Z,
  alpha,
  beta,
  phi,
  sigma.a,
  sigma.b,
  id.levels = NULL,
  sim.seed = NULL
)
```

Arguments

`ncellsper`	Vector giving the number of cells per individual. Length of the vector is taken as the number of individuals.
`X`	Covariate matrix (without intercept) for the (conditional) mean model.
`Z`	Covariate matrix (without intercept) for the zero-inflation model.
`alpha`	Column vector of true parameters from the zero-inflation model. Number of rows must match number of columns in Z.
`beta`	Column vector of true parameters from the (conditional) mean model. Number of rows must match number of columns in X.
`phi`	Overdispersion parameter for the negative binomial distribution (see details for more about parameterization).
`sigma.a`	Standard deviation for the zero-inflation model random intercept.
`sigma.b`	Standard deviation for the (conditional) mean random intercept.
`id.levels`	Individual-level IDs. If NULL set as 1,2,... up to the number of individuals.
`sim.seed`	Random seed to be used. If NULL one will be randomly chosen.

Value

Y Simulated counts

X Covariate matrix (without intercept) for the (conditional) mean model.

Z Covariate matrix (without intercept) for the zero-inflation model.

a Random effects for the zero-inflation model.

b Random effects for the (conditional) mean model.

alpha Column vector of true parameters from the zero-inflation model. Number of rows must match number of columns in Z.

beta Column vector of true parameters from the (conditional) mean model. Number of rows must match number of columns in X.

phi Overdispersion parameter for the negative binomial distribution (see details for more about parameterization).

sigma.a Standard deviation for the zero-inflation model random intercept.

sigma.b Standard deviation for the (conditional) mean random intercept.

nind Number of individuals.

ncellsper Vector giving the number of cells per individual.

id.levels Individual-level IDs.

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
```

```
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
rownames(sim_dat)<-paste("Gene",1:2)
```

`test.vc.twosigma` *Convenient wrapper function for performing (joint) likelihood ratio tests of variance components using the TWO-SIGMA model.*

Description

Convenient wrapper function for performing (joint) likelihood ratio tests of variance components using the TWO-SIGMA model.

Usage

```
test.vc.twosigma(
  count_matrix,
  mean_covar,
  zi_covar,
  mean_re = TRUE,
  zi_re = TRUE,
  id,
  return_full_fits = TRUE,
  adhoc = FALSE,
  adhoc_thresh = 0.1,
  silent = FALSE,
  disp_covar = NULL,
  weights = rep(1, ncol(count_matrix)),
  control = glmmTMBControl(),
  ncores = 1,
  cluster_type = "Fork",
  chunk_size = 1,
  lb = FALSE
)
```

Arguments

`count_matrix`	Matrix of non-negative integer read counts, with rows corresponding to genes and columns corresponding to cells. It is recommended to make the rownames the gene names for better output.
`mean_covar`	Covariates for the (conditional) mean model. Must be a matrix (without an intercept column) or a vector if a single covariate is being tested.
`zi_covar`	Covariates for the zero-inflation model. Must be a matrix (without an intercept column) or a vector if a single covariate is being tested.
`mean_re`	Should random intercepts be tested in the (conditional) mean model?
`zi_re`	Should random intercepts be tested in the zero-inflation model?
`id`	Vector of individual-level ID's. Used for random effect prediction and the adhoc method but required regardless.
`return_full_fits`	If TRUE, fit objects of class glmmTMB are returned. If FALSE, only objects of class summary.glmmTMB are returned. The latter require a much larger amount of memory to store.
`adhoc`	Should the adhoc method be used by default to judge if random effects are needed?
`adhoc_thresh`	Value below which the adhoc p-value is deemed significant (and thus RE are deemed necessary). Only used if adhoc==TRUE.
`silent`	If TRUE, progress is not printed.
`disp_covar`	Covariates for a log-linear model for the dispersion. Either a matrix or = 1 to indicate an intercept only model.
`weights`	weights, as in glm. Defaults to 1 for all observations and no scaling or centering of weights is performed. See `?glmmTMBControl`.
`control`	Control parameters for optimization in glmmTMB.
`ncores`	Number of cores used for parallelization. Defaults to 1, meaning no parallelization of any kind is done.
`cluster_type`	Whether to use a "cluster of type "Fork" or "Sock". On Unix systems, "Fork" will likely improve performance. On Windows, only "Sock" will actually result in parallelized computing.
`chunk_size`	Number of genes to be sent to each parallel environment. Parallelization is more efficient, particularly with a large count matrix, when the count matrix is 'chunked' into some common size (e.g. 10, 50, 200). Defaults to 10.
`lb`	Should load balancing be used for parallelization? Users will likely want to set to FALSE for improved performance.

Value

A list with the following elements:

- `fit_null`: Model fits under the null hypothesis. If `return_summary_fits=TRUE`, returns a list of objects of class `summary.glmmTMB`. If `return_summary_fits=FALSE`, returns a list of model fit objects of class `glmmTMB`. In either case, the order matches the row order of `count_matrix`, and the names of the list elements are taken as the rownames of `count_matrix`.
- `fit_alt`: Model fits under the alt hypothesis of the same format as `fit_null`.
- `LR_stat`: Vector of Likelihood Ratio statistics. A value of 'NA' implies a convergence issue or other model fit problem.
- `LR_p.val`: Vector of Likelihood Ratio p-values. A value of 'NA' implies a convergence issue or other model fit problem.

Details

If either model fails to converge, or the LR statistic is negative, both the statistic and p-value are assigned as NA.

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
```

```
rownames(sim_dat)<-paste("Gene",1:2)

# Run test.vc.twosigma

test.vc.twosigma(sim_dat[1,,drop=FALSE],mean_covar = X,zi_covar=Z
,mean_re = TRUE,zi_re=FALSE,id = id)
```

---

twosigma *Fit the TWO-SIGMA Model.*

Description

Fit the TWO-SIGMA Model.

Usage

```
twosigma(
  count_matrix,
  mean_covar,
  zi_covar,
  mean_re = TRUE,
  zi_re = TRUE,
  id,
  adhoc = TRUE,
  adhoc_thresh = 0.1,
  return_summary_fits = TRUE,
  disp_covar = NULL,
  weights = rep(1, ncol(count_matrix)),
  control = glmmTMBControl(),
  ncores = 1,
  cluster_type = "Fork",
  chunk_size = 10,
  lb = FALSE
)
```

Arguments

`count_matrix`	Matrix of non-negative integer read counts, with rows corresponding to genes and columns corresponding to cells. It is recommended to make the rownames the gene names for better output.
`mean_covar`	Covariates for the (conditional) mean model. Must be a matrix (without an intercept column) or = 1 to indicate an intercept only model.
`zi_covar`	Covariates for the zero-inflation model. Must be a matrix (without an intercept column), = 1 to indicate an intercept only model, or = 0 to indicate no zero-inflation model desired.
`mean_re`	Should random intercepts be included in the (conditional) mean model? Ignored if adhoc=TRUE.

`zi_re`	Should random intercepts be included in the zero-inflation model? Ignored if adhoc=TRUE.
`id`	Vector of individual-level ID's. Used for random effect prediction and the adhoc method but required regardless.
`adhoc`	Should the adhoc method be used by default to judge if random effects are needed?
`adhoc_thresh`	Value below which the adhoc p-value is deemed significant (and thus RE are deemed necessary). Only used if adhoc==TRUE.
`return_summary_fits`	If TRUE, the package returns a `summary.glmmTMB` object for each gene. If FALSE, an object of class `glmmTMB` is returned for each gene. The latter requires far more memory to store.
`disp_covar`	Covariates for a log-linear model for the dispersion. Either a matrix of covariates or = 1 to indicate an intercept only model. Random effect terms are not permitted in the dispersion model. Defaults to NULL for constant dispersion.
`weights`	weights, as in glm. Defaults to 1 for all observations and no scaling or centering of weights is performed.
`control`	Control parameters for optimization in `glmmTMB`. See `?glmmTMBControl`.
`ncores`	Number of cores used for parallelization. Defaults to 1, meaning no parallelization of any kind is done.
`cluster_type`	Whether to use a "cluster of type "Fork" or "Sock". On Unix systems, "Fork" will likely improve performance. On Windows, only "Sock" will actually result in parallelized computing.
`chunk_size`	Number of genes to be sent to each parallel environment. Parallelization is more efficient, particularly with a large count matrix, when the count matrix is 'chunked' into some common size (e.g. 10, 50, 200). Defaults to 10.
`lb`	Should load balancing be used for parallelization? Users will likely want to set to FALSE for improved performance.

Value

A list with the following elements: ##'

- `fit`: If `return_summary_fits=TRUE`, returns a list of model fit objects of class `summary.glmmTMB`. If `return_summary_fits=FALSE`, returns a list of model fit objects of class `glmmTMB`. In either case, the order matches the row order of `count_matrix`, and the names of the list elements are taken as the rownames of `count_matrix`.
- `adhoc_include_RE`: Logical vector indicator whether the adhoc method determined random effects needed. If `adhoc=F`, then a vector of NA's.
- `gene_error`: Vector indicating whether the particular gene produced an error during model fitting (TRUE) or not (FALSE).

Details

If adhoc=TRUE, any input in mean_re and zi_re will be ignored.

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
rownames(sim_dat)<-paste("Gene",1:2)

# Run twosigma

twosigma(sim_dat[1:2,],mean_covar = X,zi_covar=1,id = id)
```

`twosigmag` *Gene set testing for single-cell RNA-sequencing data adjusting for inter-gene correlation.*

Description

Gene set testing for single-cell RNA-sequencing data adjusting for inter-gene correlation.

Usage

```
twosigmag(
  count_matrix,
  index_test,
  index_ref = NULL,
  all_as_ref = FALSE,
  mean_form,
  zi_form,
  mean_form_null = NULL,
  zi_form_null = NULL,
  id,
  statistic,
  lr.df = NULL,
  covar_to_test = NULL,
  contrast_matrix = NULL,
  factor_name = NULL,
  rho = NULL,
  allow_neg_corr = FALSE,
  return_summary_fits = FALSE,
  weights = NULL,
  control = glmmTMBControl(),
  ncores = 1,
  cluster_type = "Fork",
  chunk_size = 10,
  lb = FALSE
)
```

Arguments

`count_matrix`	Matrix of non-negative integer read counts. It is recommended to make the rownames the gene names for better output. No missing values can be present in the data.
`index_test`	List of indices corresponding to rows of the count matrix that are in the test set. Names of each list element (i.e. Gene Set Names) are carried forward to output if present.
`index_ref`	List of indices corresponding to rows of the count matrix that are in the reference set. If `NULL`, a reference set is randomly selected of the same size as the test size using genes not in the test set (if `all_as_ref=FALSE`) or using all other genes (if `all_as_ref=TRUE`). See `all_as_ref`. Must be either `NULL` or a list with the same length as `index_test`.
`all_as_ref`	Should all genes not in the test set be used as the reference? If `FALSE`, a random subset is taken of size equal to the test size.
`mean_form`	Two-sided model formula for the (conditional) mean model. Formula is passed directly into glmmTMB with random effects specified as in the lme4 package. Users should ensure that the LHS of the formula contains '`count`'.
`zi_form`	One-sided model formula for the zero-inflation model under the alternative. Formula is passed directly into glmmTMB with random effects specified as in the lme4 package.
`mean_form_null`	Two-sided model formula for the (conditional) mean model under the null. Needed if and only if `statistic='LR'`. Syntax is as in `mean_form`. Users should ensure that the LHS of the formula contains '`count`'.

`zi_form_null`	One-sided model formula for the zero-inflation model under the null. Needed if and only if `statistic='LR'`. Syntax is as in `zi_form`.
`id`	Vector of individual-level (sample-level) ID's. Used to estimate inter-gene correlation and random effect prediction (if present) and is currently required.
`statistic`	Which gene-level statistic should be used. Options are Likelihood Ratio ("LR", default), Z-statistic from the mean model ("Z"),the Stouffer's method combined Z-statistic ("Stouffer"), or a contrast of regression parameters ("contrast"). If "Stouffer", covar_to_test must be in both components. If "contrast", covar_to_test is not used and must be `NULL`.
`lr.df`	degrees of freedom for the asymptotic chi-square approximation to the likelihood ratio statistic. Needed if and only if `statistic='LR'`.
`covar_to_test`	Covariate used for reporting direction (as Up or Down) of the test set and for collecting gene-level statistics. Either a string indicating the name of the covariate to use or an integer giving its associated position in the RHS of the mean_form argument. If a string, the name is matched to the predictors of the mean model, so users should ensure such a match would be unique. Not required and should be `NULL` if `statistic='contrast'`.
`contrast_matrix`	Matrix of contrasts of regression parameters from the mean model to be tested. Each row will have separate gene-level and set-level statistics. Rownames of `contrast_matrix` should correspond to a meaningful name of the hypothesis for nicely formatted output. If testing a factor, must have a number of columns exactly equal to the number of levels of the factor. Otherwise, must have one column per parameter in the mean model (including a column for the intercept.)
`factor_name`	Name of the factor being tested by `contrast_matrix`. Needed if and only if `statistic='contrast'` and `contrast_matrix` is testing a factor variable in the mean model.
`rho`	Inter-gene correlation value. If `NULL` (default), estimated using TWO-SIGMA model residuals.
`allow_neg_corr`	Should negative correlation values be allowed? If FALSE, negative correlations are set to zero (leads to conservative inference)..
`return_summary_fits`	If `TRUE`, returns a list containing objects of class `summary.glmmTMB` for each gene.
`weights`	weights, as in `glm`. Defaults to 1 for all observations and no scaling or centering of weights is performed.
`control`	Control parameters for optimization in `glmmTMB`. See `?glmmTMBControl`.
`ncores`	Number of cores used for parallelization. Defaults to 1, meaning no parallelization of any kind is done.
`cluster_type`	Whether to use a "cluster of type "Fork" or "Sock". On Unix systems, "Fork" will likely improve performance. On Windows, only "Sock" will actually result in parallelized computing.
`chunk_size`	Number of genes to be sent to each parallel environment. Parallelization is more efficient, particularly with a large count matrix, when the count matrix is 'chunked' into some common size (e.g. 10, 50, 200). Defaults to 10.

`lb`	Should load balancing be used for parallelization? Users will likely want to set to FALSE for improved performance.

Value

A list with the following elements: ##'

- `stats_gene_level_all`: Gives all gene-level statistics. Order matches the order of the inputted count matrix.
- `p.vals_gene_level`: Gives raw (unadjusted) p-values associated with `stats_gene_level_all`.
- `set_p.val`: Unadjusted set-level p-values. Order matches the order of inputted test sets.
- `set_p.val_FDR`: FDR-corrected (using the Benjamini-Hochberg procedure) set-level p-values. Order matches the order of inputted test sets.
- `estimates_gene_level`: Gives the average logFC or contrast estimate for each gene.
- `se_gene_level`: Standard error of the gene-level logFC values. Useful to construct gene-level summary statistics.
- `estimates_set_level`: Gives the set-level average of the gene-level logFC or contrast estimates.
- `direction`: Reports whether the test set tends to be Up or Down Regulated based on the sign of `estimates_set_level`.
- `corr`: Vector of estimated inter-gene correlations for each test set. Order matches the order of inputted test sets.
- `gene_level_loglik`: Vector of log-likelihood values for each gene. Values of NA indicates a model fitting or convergence problem for that gene.
- `gene_error`: Vector indicating whether the particular gene produced an error during model fitting (TRUE) or not (FALSE).
- `test_sets`: Vector of numeric indices corresponding to genes in each test set.
- `ref_sets`: Vector of numeric indices corresponding to the genes in each reference set.
- `gene_summary_fits`: Summary.glmmTMB objects for each gene from the alternative model (if return_summary_fits=TRUE)

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
```

```
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data, half under null half under alternative

sim_dat<-matrix(nrow=4,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
 if(i<2){# Gene Sets Under the Null
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
 }else{# Gene Sets Under the Alternative
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
 }
}
rownames(sim_dat)<-paste("Gene",1:4)

# Run twosigmag

twosigmag(sim_dat,index_test = list(c(1,3)),all_as_ref = TRUE,mean_form = count~X
,zi_form = ~0,id=id,covar_to_test  = "t2d_sim",statistic = "Z")
```

`twosigma_custom` *Fit the TWO-SIGMA model with custom user-specified model formulas.*

Description

Fit the TWO-SIGMA model with custom user-specified model formulas.

Usage

```
twosigma_custom(
  count_matrix,
  mean_form,
  zi_form,
  id,
  return_summary_fits = TRUE,
  silent = FALSE,
  disp_covar = NULL,
  weights = rep(1, ncol(count_matrix)),
  control = glmmTMBControl(),
  ncores = 1,
  cluster_type = "Fork",
  chunk_size = 10,
  lb = FALSE,
  internal_call = FALSE
)
```

Arguments

`count_matrix`	Matrix of non-negative integer read counts, with rows corresponding to genes and columns corresponding to cells. It is recommended to make the rownames the gene names for better output.
`mean_form`	Custom two-sided model formula for the (conditional) mean model. Formula is passed directly into glmmTMB with random effects specified as in the lme4 package. Users should ensure that the LHS of the formula begins with "count."
`zi_form`	Custom one-sided model formula for the zero-inflation model. Formula is passed directly into glmmTMB with random effects specified as in lme4.
`id`	Vector of individual-level (sample-level) ID's. Used for random effect prediction but required regardless of their presence in the model.
`return_summary_fits`	If TRUE, the package returns a `summary.glmmTMB` object for each gene. If FALSE, a `glmmTMB` object is returned for each gene. The latter requires far more storage space.
`silent`	If TRUE, progress is not printed.
`disp_covar`	Covariates for a log-linear model for the dispersion. Either a matrix of covariates or = 1 to indicate an intercept only model. Random effect terms are not permitted in the dispersion model.
`weights`	weights, as in glm. Defaults to 1 for all observations and no scaling or centering of weights is performed.
`control`	Control parameters for optimization in `glmmTMB`. See `?glmmTMBControl`.
`ncores`	Number of cores used for parallelization. Defaults to 1, meaning no parallelization of any kind is done.
`cluster_type`	Whether to use a "cluster of type "Fork" or "Sock". On Unix systems, "Fork" will likely improve performance. On Windows, only "Sock" will actually result in parallelized computing.
`chunk_size`	Number of genes to be sent to each parallel environment. Parallelization is more efficient, particularly with a large count matrix, when the count matrix is 'chunked' into some common size (e.g. 10, 50, 200). Defaults to 10.
`lb`	Should load balancing be used for parallelization? Users will likely want to set to FALSE for improved performance.
`internal_call`	Not needed by users called `twosigma_custom` directly.

Value

A list with the following elements:

- `fit`: If `return_summary_fits=TRUE`, returns a list of model fit objects of class `summary.glmmTMB`. If `return_summary_fits=FALSE`, returns a list of model fit objects of class `glmmTMB`. In either case, the order matches the row order of `count_matrix`, and the names of the list elements are taken as the rownames of `count_matrix`.
- `gene_error`: Vector indicating whether the particular gene produced an error during model fitting (TRUE) or not (FALSE).

Details

This function is likely only needed if users wish to include random effect terms beyond random intercepts. Users should be confident in their abilities to specify random effects using the syntax of lme4.

Examples

```
# Set Parameters to Simulate Some Data

nind<-10;ncellsper<-rep(50,nind)
sigma.a<-.5;sigma.b<-.5;phi<-.1
alpha<-c(1,0,-.5,-2);beta<-c(2,0,-.1,.6)
beta2<-c(2,1,-.1,.6)
id.levels<-1:nind;nind<-length(id.levels)
id<-rep(id.levels,times=ncellsper)
sim.seed<-1234

# Simulate individual level covariates

t2d_sim<-rep(rbinom(nind,1,p=.4),times=ncellsper)
cdr_sim<-rbeta(sum(ncellsper),3,6)
age_sim<-rep(sample(c(20:60),size=nind,replace = TRUE),times=ncellsper)

# Construct design matrices

Z<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(Z)<-c("t2d_sim","age_sim","cdr_sim")
X<-cbind(scale(t2d_sim),scale(age_sim),scale(cdr_sim))
colnames(X)<-c("t2d_sim","age_sim","cdr_sim")

# Simulate Data

sim_dat<-matrix(nrow=2,ncol=sum(ncellsper))
for(i in 1:nrow(sim_dat)){
   sim_dat[i,]<-simulate_zero_inflated_nb_random_effect_data(ncellsper,X,Z,alpha,beta2
   ,phi,sigma.a,sigma.b,id.levels=NULL)$Y
}
rownames(sim_dat)<-paste("Gene",1:2)

# Run twosigma_custom

twosigma_custom(sim_dat[1:2,],mean_form = count~X,zi_form = ~0,id=id)
```

Index