Package ‘MAST’

May 27, 2024

Type Package

Title Model-based Analysis of Single Cell Transcriptomics

Version 1.30.0

Date 2023-04-20

VignetteBuilder knitr

Imports Biobase, BiocGenerics, S4Vectors, data.table, ggplot2, plyr, stringr, abind, methods, parallel, reshape2, stats, stats4, graphics, utils, SummarizedExperiment(>= 1.5.3), progress, Matrix

Depends SingleCellExperiment (>= 1.2.0), R(>= 3.5)

Suggests knitr, rmarkdown, testthat, lme4(>= 1.0), blme, roxygen2(> 6.0.0), numDeriv, car, gdata, lattice, GGally, GSEABase, NMF, TxDb.Hsapiens.UCSC.hg19.knownGene, rsvd, limma, RColorBrewer, BiocStyle, scater, DelayedArray, HDF5Array, zinbwave, dplyr

Description Methods and models for handling zero-inflated single cell assay data.

License GPL(>= 2)


RoxygenNote 7.2.3

LazyData true

biocViews GeneExpression, DifferentialExpression, GeneSetEnrichment, RNASeq, Transcriptomics, SingleCell

BugReports https://github.com/RGLab/MAST/issues
URL: https://github.com/RGLab/MAST/

Encoding: UTF-8

**git_url** https://git.bioconductor.org/packages/MAST

**git_branch** RELEASE_3_19

**git_last_commit** da42c5a

**git_last_commit_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-26

**Author** Andrew McDavid [aut, cre],
Greg Finak [aut],
Masanao Yajima [aut]

**Maintainer** Andrew McDavid <Andrew_McDavid@urmc.rochester.edu>

Contents

MAST-package ............................................. 4
applyFlat .................................................. 4
BayesGLMlike-class .................................... 5
calcZ ....................................................... 5
colData<-.SingleCellAssay,DataFrame-method .......... 6
collectResiduals .......................................... 7
computeEtFromCt .......................................... 8
convertMASTClassicToSingleCellAssay ................ 9
CovFromBoots ............................................ 10
defaultPrior ............................................ 11
dof ......................................................... 11
Drop ....................................................... 12
ebayes .................................................... 12
expavg ..................................................... 13
filterLowExpressedGenes ................................ 14
fit ......................................................... 14
freq ....................................................... 15
FromFlatDF ............................................... 16
FromMatrix ............................................... 17
getConcordance ......................................... 18
getwellKey ............................................... 20
GLMlike-class .......................................... 20
gseaAfterBoot ........................................... 21
GSEATests-class ......................................... 23
hushWarning ............................................. 23
Hypothesis ............................................... 24
impute ..................................................... 24
influence.bayesglm ...................................... 25
invlogit .................................................. 26
LMERlike-class ......................................... 26
**Description**

Methods for analysing single cell assay data using hurdle models.

**Details**

This package provides data structures and functions for statistical analysis of single-cell assay data such as Fluidigm single cell gene expression assays.

**Author(s)**

**Maintainer:** Andrew McDavid <Andrew_McDavid@urmc.rochester.edu>

Authors:

- Greg Finak <gfinak@fredhutch.org>
- Masanao Yajima <myajima@fredhutch.org>

**References**


**See Also**

Useful links:

- [https://github.com/RGLab/MAST/](https://github.com/RGLab/MAST/)
- Report bugs at [https://github.com/RGLab/MAST/issues](https://github.com/RGLab/MAST/issues)

---

**applyFlat**

*Apply a vectorized binary operation recycling over last dimension*

**Description**

When `x` is an array of order `K`, and `y` is an array of order `K-1`, whose dimensions otherwise agree, apply `FUN` by recycling `y` as necessary over dimension `K` of `x`.

**Usage**

```r
applyFlat(x, y, FUN = "-")
```
BayesGLMlike-class

Arguments

- **x**: array, order K
- **y**: array, order K-1
- **FUN**: vectorized binary operation

Value

array, order K equal to FUN(x,y)

Examples

```r
## Dumb example, could be done with scale(..., scale=FALSE)
x0 = matrix(1:10, nrow=2)
y0 = rowMeans(x0)
dim(y0) = c(1, 2)
x1 = MAST:::applyFlat(x0, y0)
stopifnot(rowMeans(x1) == 0)
```

BayesGLMlike-class  Wrapper for bayesian GLM

Description

Wrapper for bayesian GLM

Slots

- **prior numeric**: optional 3d array used to specify prior for coefficients
- **useContinuousBayes logical**: should bayesglm be used to fit the continuous component as well?

calcZ  Get Z or T statistics and P values after running gseaAfterBoot

Description

The Z or T statistics may be reported by component (discrete/continuous) when combined='no' or combined by Fisher's or Stouffer's method (combined='fisher' or combined='stouffer'). Fisher’s method uses the product of the p-values, while Stouffer’s method uses the sum of the Z/T scores. The "Z" score returned by Fisher is the normal quantile that would yield the observed Fisher P-value, whose sign is derived from the sign of the maximum component Z score. The "Z" score returned by Stouffer when testType='normal' is the sum of the Z scores, over sqrt(2). When testType='t' it is a weighted combination of the Z scores, with weights corresponding to the degrees of freedom in each of the t statistics. A t-approximation to this sum of t-variables is derived by matching moments. It seems to be fairly accurate in practice.
Usage

calcZ(gseaObj, testType = "t", combined = "none")

Arguments

gseaObj output from gseaAfterBoot
testType either 'normal' or 't'. The 't' test adjusts for excess kurtosis due to the finite number of bootstrap replicates used to estimate the variance of the statistics. This will result in more conservative inference.
combined character one of 'none', 'fisher' or 'stouffer'

Value

3D array with dimensions set (modules) comp ('continous' or 'disc'rete) and metric ('Z' stat and two sided 'P' value that $P(|Z|)$) if combined='no', otherwise just a matrix.

See Also
gseaAfterBoot

Examples

### See the examples in gseaAfterBoot
example(gseaAfterBoot)

---

**Description**

Replace colData with a DataFrame. Checks to make sure that row.names(value) match colnames(x), in contrast to the parent method Checks for a wellKey column, as well.

Usage

```r
## S4 replacement method for signature 'SingleCellAssay,DataFrame'
colData(x) <- value
```

Arguments

- **x** SingleCellAssay
- **value** DataFrame

Value

modified SingleCellAssay
Description

After each gene is fit, a hook function can optionally be run and the output saved. This allows extended computations to be done using the fitted model, without keeping it in memory. Here this is used to calculate various residuals, though in some cases they can be done using only the information contained in the ZlmFit-class.

Usage

```r
collectResiduals(x, sca, newLayerName = "Residuals")
discrete_residuals_hook(x)
continuous_residuals_hook(x)
combined_residuals_hook(x)
deviance_residuals_hook(x)
fitted_phat(x)
partialScore(x, effectRegex)
```

Arguments

- `x` ZlmFit-class
- `sca` SingleCellAssay object to which the residuals should be added
- `newLayerName` character name of the assay layer
- `effectRegex` a regular expression naming columns of the design corresponding to $Z_0$. Generally these should be the treatment effects of interest.

Value

copy of sca with new layer

Functions

- `discrete_residuals_hook()`: Hook to get the discrete residuals, ie, difference between expected probability of expression and observed
- `continuous_residuals_hook()`: Hook to get the continuous residuals, ie, residuals for conditionally positive observations. If an observation is zero, it’s residual is defined to be zero as well.
- `combined_residuals_hook()`: Hook to get the combined residuals, ie, $Y - E(U) * E(V)$
• `deviance_residuals_hook()`: Standardized deviance residuals hook. Computes the sum of the standardized deviance residuals for the discrete and continuous models (scaled to have unit variance). If the observation is zero then only the discrete component is used.

• `fitted_phat()`: Hook to return \( p_\text{hat} \), the predicted probability of expression.

• `partialScore()`: Compute \( Y_i - E(V_i|X_i, Z_0)E(U|X_i, Z_0) \), where \( Z_0 \) is a treatment effect (being left in) and \( X_i \) is a nuisance effect (being regressed out).

### Total residual types

Each component of the model contributes several flavors of residual, which can be combined in various fashions. The discrete residual can be on the response scale (thus subtracting the predicted probability of expression from the 0/1 expression value). Or it can be a deviance residual, revealing something about the log-likelihood.

### Partial residuals

It’s also possible to consider partial residuals, in which the contribution of a particular covariate is added back into the model.

### See Also

zlm

### Examples

```r
data(vbetaFA)
svbeta <- subset(vbetaFA, ncells==1)
svbeta <- svbeta[freq(svbeta)>0.4]
window <- function(x1) lapply(assays(x1), function(x2) x2[1:3, 1:6])
# total residuals of the response
z1 <- zlm(~ Stim.Condition, svbeta, hook=discrete_residuals_hook)
window(collectResiduals(z1, svbeta))
z2 <- zlm(~ Stim.Condition, svbeta, hook=continuous_residuals_hook)
window(collectResiduals(z2, svbeta))
z3 <- zlm(~ Stim.Condition, svbeta, hook=combined_residuals_hook)
window(collectResiduals(z3, svbeta))
# partial residuals
colData(svbeta)$ngeneson <- colMeans(assay(svbeta)>0)
z5 <- zlm(~ Stim.Condition + ngeneson, svbeta)
partialScore(z5, 'Stim.Condition')
```

---

**computeEtFromCt**

**Compute the Et from the Ct**

### Description

Computes the Et value from the Ct value in an existing data frame and returns a new data frame with the Et column appended.
computeEtFromCt(df, column = "Ct", Cmax = 40)

Arguments

- df: a data.frame
- column: The name of the Ct column. A character. 'Ct' by default.
- Cmax: the maximum number of cycles performed. 40 by default.

Value

A copy of df with the 'Et' column appended

Author(s)

Greg Finak

Examples

data(vbeta)
vbeta <- computeEtFromCt(vbeta)

convertMASTClassicToSingleCellAssay

Convert a MASTClassic SingleCellAssay

Description

Convert a SingleCellAssay object created with the MASTClassic package to an object recognized by the new MAST package

Usage

convertMASTClassicToSingleCellAssay(object = NULL)

Arguments

- object: of class SingleCellAssay created by MASTClassic

Details

The function will extract the relevant information from the attributes of the old object and construct a new SingleCellAssay that is recognized by MAST. This function checks that the object is a MASTClassic SingleCellAssay object. It will stop if it is not a SingleCellAssay, return a converted SingleCellAssay if object was created by MASTClassic, and return the original object if the object is already compatible.
Value

A MAST SingleCellAssay object.

Note

Type checking for old object is not performed.

Examples

data(vbetaFA)
convertMASTClassicToSingleCellAssay(vbetaFA)

Description

Extract the inter-gene covariance matrices for continuous and discrete components of a MAST model for a given coefficient from bootstrap replicates.

Usage

CovFromBoots(boots = NULL, coefficient = NULL)

Arguments

- `boots`: a multidimensional array returned by `bootVcov1()` or `pbootVcov1()`.
- `coefficient`: 'character' the name of the model coefficient for which to return the inter-gene covariance matrices.

Value

list with components "C" and "D" containing covariance matrices for the continuous and discrete components of the model.
defaultPrior

Initialize a prior to be used a prior for BayeGLMlike/BayesGLMlike2

Description

Initialize a prior to be used a prior for BayeGLMlike/BayesGLMlike2

Usage

defaultPrior(names)

Arguments

names character vector of coefficients. The ‘(Intercept)’ will be ignored.

Value

3d array, with leading dimension giving the prior ‘loc’ation, ‘scale’ and degrees of freedom (df), second dimension giving the component (‘C’ontinuous or ‘D’iscrete) and trailing dimension giving the coefficient to which the prior applies. The location is initialized to be 0, the scale to 2, and degrees of freedom of 1, following the default of bayesglm.

Examples

dp <- defaultPrior('Stim.ConditionUnstim')
## Not run:
data(vbetaFA)
zlmVbeta <- zlm(~ Stim.Condition, vbetaFA, method='bayesglm', coefPrior=dp)
## End(Not run)

dof

Degrees of freedom of Zero inflated model

Description

Degrees of freedom of Zero inflated model

Usage

dof(object)

Arguments

object LMlike or subclass
Value

vector giving the model degrees of freedom for continuous and discrete

---

Drop

*Drop specified dimension from an array*

Description

Like drop(x) but only dropping specified dimensions. There is no testing that the specified dimensions are actually singletons.

Usage

```r
Drop(x, d)
```

Arguments

- `x` array of at least d dimensions
- `d` dimension(s) to drop

Value

array x

Examples

```r
x = array(1:4, dim=c(1, 2, 1, 2))
dx = MAST:::Drop(x, 1)
stopifnot(all(dim(dx)==c(2,1,2)))
```

ebayes

*Estimate hyperparameters for hierarchical variance model for continuous component*

Description

ebayesControl is a named list with (optional) components 'method' (one of 'MOM' or 'MLE') and 'model' (one of 'H0' or 'H1') method MOM uses a method-of-moments estimator, while MLE using the marginal likelihood. H0 model estimates the precisions using the intercept alone in each gene, while H1 fits the full model specified by `mm`

Usage

```r
ebayes(assay_t, ebayesControl, mm, truncate = Inf)
```
expavg

Arguments

- **assay_t**: cells X genes matrix
- **ebayesControl**: list with (optional) components 'method', 'model'. See details.
- **mm**: a model matrix, used when model='H1'.
- **truncate**: Genes with sample precisions exceeding this value are discarded when estimating the hyper parameters

Value

numeric of length two, giving the hyperparameters in terms of a variance (\(v\)) and prior observations (\(df\)), inside a structure, with component hess, giving the Fisher Information of the hyperparameters.

---

expavg  

**Exponential average**

Description

Puts log transformed values onto natural scale and takes mean of vector. Calculates \(\text{mean}(2^x - 1)\)

Usage

expavg(x)

Arguments

- **x**: numeric

Value

numeric

Examples

```r
x <- 1:10
logmean(expavg(x))
```
filterLowExpressedGenes

Filter low-expressing genes

Description
Filter out genes that have less than some percent threshold expression across all libraries

Usage
filterLowExpressedGenes(assay, threshold = 0.1)

Arguments
assay a SingleCellAssay object
threshold a numeric between 0, and 1, specifying the threshold frequency below which
genpes will be filtered out

Value
SingleCellAssay

Examples
data(vbetaFA)
filterLowExpressedGenes(vbetaFA)

fit
fit a zero-inflated regression

Description
Given a design and formula, fit the zero inflated regression, storing the fits in slots fitC and fitD

Usage
fit(object, response, ...)

## S4 method for signature 'LMERlike,missing'
fit(object, response, silent = TRUE, ...)

Arguments
object inheriting from LMLike
response a vector, same length as the design, or if missing then use the current response
... currently ignored
silent mute some warnings emitted from the underlying modeling functions
freq

Value
LMlike or subclass

Description
freq returns the frequency of expression, i.e., the proportion of non-zero values in sc. NAs can be optionally removed.

Usage
freq(sc, na.rm = TRUE)
condmean(sc)
condSd(sc)
umexp(sc)

Arguments
sc SingleCellAssay
na.rm should NAs be removed, or carried through?

Value
vector of proportions

Functions
• condmean(): Report the mean non-zero expression value for each gene. NAs are always removed.
• condSd(): Report standard deviation of expression, for positive et for each gene. NAs are always removed.
• numexp(): Report number of expressing cells ($>0$) per gene. NAs are removed.

Examples
data(vbetaFA)
freq(vbetaFA)
condmean(vbetaFA)
Construct a `SingleCellAssay` (or derived subclass) from a ‘flat’ (melted) `data.frame/data.table`

**Description**

`SingleCellAssay` are a generic container for such data and are simple wrappers around `SummarizedExperiment` objects. Subclasses exist that embue the container with additional attributes, eg `FluidigmAssay`.

**Usage**

```r
FromFlatDF(
  dataframe,
  idvars,
  primerid,
  measurement,
  id = numeric(0),
  cellvars = NULL,
  featurevars = NULL,
  phenovars = NULL,
  class = "SingleCellAssay",
  check_sanity = TRUE,
  ...
)
```

**Arguments**

- `dataframe`: A ‘flattened’ `data.frame` or `data.table` containing columns giving cell and feature identifiers and a measurement column
- `idvars`: character vector naming columns that uniquely identify a cell
- `primerid`: character vector of length 1 that names the column that identifies what feature (i.e. gene) was measured
- `measurement`: character vector of length 1 that names the column containing the measurement
- `id`: An identifier (eg, experiment name) for the resulting object
- `cellvars`: Character vector naming columns containing additional cellular metadata
- `featurevars`: Character vector naming columns containing additional feature metadata
- `phenovars`: Character vector naming columns containing additional phenotype metadata
- `check_sanity`: (default: TRUE) Set FALSE to override sanity checks that try to ensure that the default assay is log-transformed and has at least one exact zero. See `defaultAssay` for details on the “default assay” which is assumed to contain log transformed data.
- `...`: additional arguments are ignored
FromMatrix

Construct a SingleCellAssay from a matrix or array of expression

Description

If the gene expression measurements are already in a rectangular form, then this function allows an easy way to construct a SingleCellAssay object while still doing some sanity checking of inputs.

Usage

```r
FromMatrix(
  exprsArray,
  cData,
  fData,
  class = "SingleCellAssay",
  check_sanity = TRUE,
  check_logged = check_sanity
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exprsArray</td>
<td>matrix, or a list of matrices, or an array. Columns are cells, rows are genes.</td>
</tr>
<tr>
<td>cData</td>
<td>cellData an object that can be coerced to a DataFrame, ie, data.frame, AnnotatedDataFrame. Must have as many rows as ncol(exprsArray)</td>
</tr>
<tr>
<td>fData</td>
<td>featureData an object that can be coerced to a DataFrame, ie, data.frame, AnnotatedDataFrame. Must have as many rows as nrow(exprsArray).</td>
</tr>
<tr>
<td>class</td>
<td>desired subclass of object. Default SingleCellAssay.</td>
</tr>
</tbody>
</table>
check_sanity  (default: TRUE) Set FALSE to override sanity checks that try to ensure that the default assay is log-transformed and has at least one exact zero. See defaultAssay for details on the "default assay" which is assumed to contain log transformed data.

check_logged   alias for check_sanity

Value

an object of class class

See Also

defaultAssay

Examples

ncells <- 10
ngenes <- 5
fData <- data.frame(primerid=LETTERS[1:ngenes])
cData <- data.frame(wellKey=seq_len(ncells))
mat <- matrix(rnorm(ncells*ngenes), nrow=ngenes)
sca <- FromMatrix(mat, cData, fData)
stopifnot(inherits(sca, 'SingleCellAssay'))
stopifnot(inherits(sca, 'SummarizedExperiment'))

# If there are mandatory keywords expected by a class, you'll have to manually set them yourself
cData$ncells <- 1
fd <- FromMatrix(mat, cData, fData)
stopifnot(inherits(fd, 'SingleCellAssay'))

getConcordance

Get the concordance between two experiments

Description

Return the concordance between two assays (i.e. single cell and hundred cell). The "average" of singleCellRef (after adjusting for the number of cells) and singleCellComp are taken per gene, per groups. A data.frame with one row per gene-groups is returned with some additional columns.

Usage

getConcordance(
  singleCellRef,
  singleCellcomp,
  groups = NULL,
  fun.natural = expavg,
  fun.cycle = logmean
)
getConcordance

getwss(concord, nexp)
getss(concord)
getrc(concord)

Arguments

- singleCellRef: "reference" SingleCellAssay
- singleCellcomp: "comparison" SingleCellAssay
- groups: character vector giving variable(s) on which the comparison is conditioned
- fun.natural: function to transform the SingleCellAssays to a mRNA proportional level
- fun.cycle: inverse function of fun.natural
- concord: data.frame returned by getConcordance
- nexp: number of expressed cells per row in concord

Value

- concordance between two assays

Functions

- getwss(): get the sum of squares, weighted by nexp
- getss(): return the sum of squares
- getrc(): Return Lin’s (1989) concordance correlation coefficient

Author(s)

- Andrew McDavid

See Also

- plotSCAConcordance

Examples

data(vbetaFA)
sca1 <- subset(vbetaFA, ncells==1)
sca100 <- subset(vbetaFA, ncells==100)
concord <- getConcordance(sca1, sca100)
getss(concord)
getrc(concord)
### getwellKey

Accessor for wellKey

**Description**

This returns the wellKey, which is a unique identifier generated by idvars in the mapping.

**Usage**

```r
getwellKey(sc)
```

**Arguments**

- `sc` An object with a wellKey

**Value**

integer giving the unique id generated

**Examples**

```r
data(vbetaFA)
getwellKey(vbetaFA)
colData(vbetaFA)$wellKey
```

---

### GLMlike-class

Wrapper for regular glm/lm

**Description**

Wrapper for regular glm/lm

**Usage**

```r
## S4 method for signature 'GLMlike'
vcov(object, which, ...)
```

**Arguments**

- `object` GLMlike
- `which` character, one of 'C', 'D'.
- `...` ignored

**Value**

covariance matrix
Methods (by generic)

• vcov(GLMlike): return the variance/covariance of component which

Slots

weightFun function to map expression values to probabilities of expression. Currently unused.

Description

Modules defined in sets are tested for average differences in expression from the "average" gene. By using bootstraps, the between-gene covariance of terms in the hurdle model is found, and is used to adjust for coexpression between genes. We drop genes if the coefficient we are testing was not estimable in original model fit in zFit or in any of the bootstrap replicates (evidenced an NA in the bootstrap array). This might yield overly conservative inference. Since bootstrapping is a randomized procedure, the degrees of freedom of a module (and its variance parameters) might differ from run-to-run. You might try setting var_estimate='modelbased' to relax this requirement by assuming independence between genes and then using the asymptotic covariance estimates, which are deterministic, but may result in overly-generous inference.

Usage

gseaAfterBoot(
  zFit,
  boots,
  sets,
  hypothesis,
  control = gsea_control(n_randomize = Inf, var_estimate = "bootall")
)

gsea_control(n_randomize = Inf, var_estimate = "bootall")

Arguments

zFit object of class ZlmFit
boots bootstraps of zFit
sets list of indices of genes
hypothesis a Hypothesis to test. Currently only one degree CoefficientHypothesis are supported.
control parameters as provided by gsea_control. See details.
n_randomize the number of genes to sample to approximate the non-module average expres-
sion. Set to Inf to turn off the approximation (the default).
var_estimate the method used to estimate the variance of the modules, one of bootall, bootdiag, or modelbased.
Value

Object of class GSEATests, containing slots tests, 4D array and bootR, the number of bootstrap replicates.

Functions

- gsea_control(): set control parameters. See Details.

control

control is a list with elements:

- n_randomize, giving the number of genes to sample to approximate the non-module average expression. Set to Inf to turn off the approximation (the default).
- var_estimate, giving the method used to estimate the variance of the modules. bootall uses the bootstrapped covariance matrices. bootdiag uses only the diagonal of the bootstrapped covariance matrix (so assuming independence across genes). modelbased assumes independence across genes and uses the variance estimated from the model.

Return Value

A 4D array is returned, with dimensions "set" (each module), "comp" ('disc'rete or 'cont'inuous), "metric" ('stat' gives the average of the coefficient, 'var' gives the variance of that average, 'dof' gives the number of genes that were actually tested in the set), "group" ('test' for the genes in test-set, "null" for all genes outside the test-set).

See Also

calcZ
summary,GSEATests-method

Examples

data(vbetaFA)
vb1 = subset(vbetaFA, ncells==1)
vb1 = vb1[,freq(vb1)>.1][1:15,]
zf = zlm(~Stim.Condition, vb1)
boots = bootVcov1(zf, 5)
sets = list(A=1:5, B=3:10, C=15, D=1:5)
gsea = gseaAfterBoot(zf, boots, sets, CoefficientHypothesis('Stim.ConditionUnstim'))
## Use a model-based estimate of the variance/covariance.
gsea_mb = gseaAfterBoot(zf, boots, sets, CoefficientHypothesis('Stim.ConditionUnstim'),
control = gsea_control(var_estimate = 'modelbased'))
calcZ(gsea)
summary(gsea)
GSEATests-class

An S4 class for Gene Set Enrichment output

Description

This holds output from a call to gseaAfterBoot. It primarily provides a summary method.

Slots

tests array: gene sets X discrete, continuous X stat, variance, degrees of freedom, avg correlation X test, null
bootR number of bootstrap replicates

See Also

gseaAfterBoot
calcZ
summary,GSEATests-method

hushWarning

Selectively muffle warnings based on output

Description

Selectively muffle warnings based on output

Usage

hushWarning(expr, regexp)

Arguments

expr an expression
regexp a regexp to be matched (with str_detect)

Value

the result of expr

Examples

hushWarning(warning('Beware the rabbit'), 'rabbit')
hushWarning(warning('Beware the rabbit'), 'hedgehog')
Hypothesis  

_Hypothesis_ can be any linear combination of coefficients, compared to zero. Specify it as a character vector that can be parsed to yield the desired equalities ala makeContrasts. A CoefficientHypothesis is a hypothesis for which terms are singly or jointly tested to be zero (generally the case in a t-test or F-test), by dropping coefficients from the model.

**Usage**

Hypothesis(hypothesis, terms)

**Arguments**

- **hypothesis**: a character vector specifying a hypothesis, following makeContrasts, or a character vector naming coefficients to be dropped.
- **terms**: an optional character vector giving the terms (column names from the model.matrix) out of which the contrasts will be contrasted. If missing then most functions will attempt to fill this in for you at run time.

**Value**

a Hypothesis with a "transformed" component

**See Also**

zlm waldTest lrTest

**Examples**

```r
h <- Hypothesis('Stim.ConditionUnstim', c('(Intercept)', 'Stim.ConditionUnstim'))
h@contrastMatrix
```

---

**impute**  

_impute missing continuous expression for plotting_

**Description**

If there are no positive observations for a contrast, it is generally not estimable. However, for the purposes of testing we can replace it with the least favorable value with respect to the contrasts that are defined.

**Usage**

impute(object, groupby)
Arguments

object  Output of predict

groupby  Variables (column names in predict) to group by for imputation (facets of the plot)

Value

data.table

Examples

##See stat_ell
eexample(stat_ell)

Description

The influence function

Usage

## S3 method for class 'bayesglm'
influence(model, do.coef = TRUE, ...)

Arguments

model  bayesglm

do.coef  see influence.glm

...  ignored

Value

see influence.glm
invlogit  
*Inverse of logistic transformation*

**Description**
Inverse of logistic transformation

**Usage**
```r
invlogit(x)
```

**Arguments**
- `x` numeric

**Value**
numeric

**Examples**
```r
x <- 1:5
invlogit(log(x/(1-x)))
```

---

**LMERlike-class**  
*Wrapper for lmer/glmer*

**Description**
A horrendous hack is employed in order to do arbitrary likelihood ratio tests: the model matrix is built, the names possibly mangled, then fed in as a symbolic formula to glmer/lmer. This is necessary because there is no (easy) way to specify an arbitrary fixed-effect model matrix in glmer.

**Usage**
```r
## S4 method for signature 'LMERlike'
update(object, formula., design, keepDefaultCoef = FALSE, ...)

## S4 method for signature 'LMERlike'
vcov(object, which, ...)

## S4 method for signature 'LMERlike'
coef(object, which, singular = TRUE, ...)

## S4 method for signature 'LMERlike'
logLik(object)
```
**LMlike-class**

**Arguments**

- **object**  
  LMERlike

- **formula**
  formula

- **design**  
  something coercible to a data.frame

- **keepDefaultCoef**
  logical. Should the coefficient names be preserved from object or updated if the model matrix has changed?

- **...**  
  In the case of vcov, ignored. In the case of update, passed to model.matrix.

- **which**
  character, one of 'C', 'D'.

- **singular**
  logical. Should NA coefficients be returned?

**Value**

see the section "Methods (by generic)"

**Methods (by generic)**

- update(LMERlike): update the formula or design matrix
- vcov(LMERlike): return the variance/covariance of component which
- coef(LMERlike): return the coefficients. The horrendous hack is attempted to be undone.
- logLik(LMERlike): return the log-likelihood

**Slots**

- **pseudoMM** part of this horrendous hack.

- **strictConvergence** logical (default: TRUE) return results even when the optimizer or *lmer complains about convergence

- **optimMsg** character record warnings from lme. NA_character_ means no warnings.

---

**LMlike-class**

*Linear Model-like Class*

**Description**

Wrapper around modeling function to make them behave enough alike that Wald tests and Likelihood ratio are easy to do. To implement a new type of zero-inflated model, extend this class. Depending on how different the method is, you will definitely need to override the fit method, and possibly the model.matrix, model.matrix<-, update, coef, vcov, and logLik methods.
Usage

```r
## S4 method for signature 'LMlike'
summary(object)

## S4 method for signature 'LMlike'
update(object, formula, design, keepDefaultCoef = FALSE, ...)

## S4 method for signature 'LMlike,CoefficientHypothesis'
waldTest(object, hypothesis)

## S4 method for signature 'LMlike,matrix'
waldTest(object, hypothesis)

## S4 method for signature 'LMlike,character'
lrTest(object, hypothesis)

## S4 method for signature 'LMlike,CoefficientHypothesis'
lrTest(object, hypothesis)

## S4 method for signature 'LMlike,Hypothesis'
lrTest(object, hypothesis)

## S4 method for signature 'LMlike,matrix'
lrTest(object, hypothesis)

## S4 method for signature 'GLMlike'
logLik(object)
```

Arguments

- **object**: `LMlike`
- **formula**: `formula`
- **design**: something coercible to a `data.frame`
- **keepDefaultCoef**: logical. Should the coefficient names be preserved from `object` or updated if the model matrix has changed?
- **...**: passed to `model.matrix`
- **hypothesis**: one of a `CoefficientHypothesis`, `Hypothesis` or contrast matrix.

Value

see section "Methods (by generic)"

Methods (by generic)

- `summary(LMlike)`: Print a summary of the coefficients in each component.
- `update(LMlike)`: update the formula or design from which the `model.matrix` is constructed
\begin{itemize}
  \item \texttt{waldTest(object = LMlike, hypothesis = CoefficientHypothesis)}: Wald test dropping single term specified by \texttt{CoefficientHypothesis} hypothesis
  \item \texttt{waldTest(object = LMlike, hypothesis = matrix)}: Wald test of contrast specified by contrast matrix hypothesis
  \item \texttt{lrTest(object = LMlike, hypothesis = character)}: Likelihood ratio test dropping entire term specified by character hypothesis naming a term in the symbolic formula.
  \item \texttt{lrTest(object = LMlike, hypothesis = CoefficientHypothesis)}: Likelihood ratio test dropping single term specified by \texttt{CoefficientHypothesis} hypothesis
  \item \texttt{lrTest(object = LMlike, hypothesis = matrix)}: Likelihood ratio test dropping single term specified by contrast matrix hypothesis
  \item \texttt{logLik(GLMlike)}: return the log-likelihood of a fitted model
\end{itemize}

\textbf{Slots}

\begin{description}
  \item[design] a data.frame from which variables are taken for the right hand side of the regression
  \item[fitC] The continuous fit
  \item[fitD] The discrete fit
  \item[response] The left hand side of the regression
  \item[fitted] A \texttt{logical} with components "C" and "D", \texttt{TRUE} if the respective component has converged
  \item[formula] A \texttt{formula} for the regression
  \item[fitArgsC] \item[fitArgsD] Both \texttt{list}s giving arguments that will be passed to the fitter (such as convergence criteria or case weights)
\end{description}

\textbf{See Also}

\begin{itemize}
  \item \texttt{coef}
  \item \texttt{lrTest}
  \item \texttt{waldTest}
  \item \texttt{vcov}
  \item \texttt{logLik}
\end{itemize}
Calculate log-fold changes from hurdle model components

Description

Using the delta method, estimate the log-fold change from a state given by a vector \( \text{contrast0} \) and the state(s) given by \( \text{contrast1} \).

Usage

\[
\text{logFC}(\text{zlmpfit}, \text{contrast0}, \text{contrast1})
\]

\[
\text{getLogFC}(\text{zlmpfit}, \text{contrast0}, \text{contrast1})
\]

Arguments

- `zlmpfit` - ZlmFit output
- `contrast0` - vector of coefficients giving baseline contrast, or a \text{Hypothesis}. If missing, then the '(Intercept)' is used as baseline.
- `contrast1` - matrix of coefficients giving comparison contrasts, or a \text{Hypothesis}. If missing, then all non-(Intercept) coefficients are compared.

Details

The log-fold change is defined as follows. For each gene, let \( u(x) \) be the expected value of the continuous component, given a covariate \( x \) and the estimated coefficients \( \text{coefC} \), ie, \( u(x) = \text{crossprod}(x, \text{coefC}) \). Likewise, let \( v(x) = 1/(1+\exp(-\text{crossprod}(-\text{coefD}, x))) \) be the expected value of the discrete component. The log fold change from \( \text{contrast0} \) to \( \text{contrast1} \) is defined as

\[ u(\text{contrast1})v(\text{contrast1}) - u(\text{contrast0})v(\text{contrast0}). \]

Note that for this to be a log-fold change, then the regression for \( u \) must have been fit on the log scale. This is returned in the matrix \text{logFC}. An approximation of the variance of \text{logFC} (applying the delta method to formula defined above) is provided in \text{varLogFC}.

Value

- list of matrices ‘logFC’ and ‘varLogFC’, giving the log-fold-changes for each contrast (columns) and genes (rows) and the estimated sampling variance thereof

Functions

- \text{getLogFC}(): Return results as a perhaps friendlier data\.table
**Caveats**

1. When `method='bayesglm'` (the default), it’s no longer necessarily true that the log fold change from condition A to B will be the inverse of the log fold change from B to A if the models are fit separately. This is due to the shrinkage in `bayesglm`.

2. The log fold change can be small, but the Hurdle p-value small and significant when the sign of the discrete and continuous model components are discordant so that the marginal log fold change cancels out. The large sample sizes present in many single cell experiments also means that there is substantial power to detect even small changes.

3. When there is no expression in a gene for a coefficient that is non-zero in either `condition0` or `condition1` we return `NA` because there is not any information to estimate the continuous component. Technically we might return plus or minus infinity, but there is not a straightforward way to estimate a confidence interval in any case. See [https://support.bioconductor.org/p/99244/](https://support.bioconductor.org/p/99244/) for details.

**See Also**

Hypothesis

summary,ZlmFit-method

**Examples**

data(vbetaFA)

zz <- zlm(~ Stim.Condition+Population, vbetaFA[1:5,])

# log-fold changes in terms of intercept (which is Stim(SEB) and CD154+VbetaResponsive)

lfcStim <- logFC(zz)

# If we want to compare against unstim, we can try the following

coeffnames <- colnames(coef(zz, 'D'))

contrast0 <- setNames(rep(0, length(coefnames)), coeffnames)

contrast0[c('Intercept', 'Stim.ConditionUnstim')] <- 1

contrast1 <- diag(length(coefnames))

rownames(contrast1) <- colnames(contrast1) <- coeffnames

contrast1['Intercept',] <- 1

lfcUnstim <- logFC(zz, contrast0, contrast1)

# log-fold change with itself is 0

stopifnot(all(lfcUnstim$logFC[,2]==0))

# inverse of log-fold change with Stim as reference

stopifnot(all(lfcStim$logFC[,1]==-lfcUnstim$logFC[,1]))

## As a data.table:

getLogFC(zz)

---

**logmean**

**Log mean**

**Description**

Takes mean of natural scaled values and then logarithm Approximately the inverse operation of `expavg` Calculates log2(mean(x) + 1)
Usage

logmean(x)

Arguments

x numeric

Value

numeric

Examples

x <- 1:10
expavg(logmean(x))

LRT Likelihood Ratio Tests for SingleCellAssays

Description

Tests for a change in ET binomial proportion or mean of positive ET Likelihood Ratio Test for SingleCellAssay objects

Usage

LRT(sca, comparison, ...)

## S4 method for signature 'SingleCellAssay,character'
LRT(sca, comparison, referent = NULL, groups = NULL, returnall = FALSE)

Arguments

sca A SingleCellAssay class object
comparison A character specifying the factor for comparison
... ignored
referent A character specifying the reference level of comparison.
groups A optional character specifying a variable on which to stratify the test. For each level of groups, there will be a separate likelihood ratio test.
returnall A logical specifying if additional rows should be returned with information about the different components of the test.

Details

Combined Likelihood ratio test (binomial and normal) for SingleCellAssay and derived objects. This function is deprecated, please use lrTest instead.
**lrTest**

**Value**

data.frame

**See Also**

zlm ZlmFit

**Examples**

data(vbetaFA)
LRT(vbetaFA, 'Stim.Condition', 'Unstim')

---

### Description

Compares the change in likelihood between the current model and one subject to contrasts tested in hypothesis. hypothesis can be one of a character giving complete factors or terms to be dropped from the model, CoefficientHypothesis giving names of coefficients to be dropped, Hypothesis giving contrasts using the symbolically, or a contrast matrix, with one row for each coefficient in the full model, and one column for each contrast being tested.

### Usage

```
lrTest(object, hypothesis, ...)
```

### Arguments

- **object**: LMlike or subclass
- **hypothesis**: the hypothesis to be tested. See details.
- **...**: optional arguments, passed to fitting functions

### Value

array giving test statistics

### See Also

fit
waldTest
Hypothesis
CoefficientHypothesis

### Examples

#see ZlmFit-class for examples
example('ZlmFit-class')
**Description**

A 3D array with first dimension being the genes, next dimension giving information about the test (the degrees of freedom, Chisq statistic, and P value), and final dimension being the value of these quantities on the discrete, continuous and hurdle (combined) levels.

**Usage**

```r
## S4 method for signature 'ZlmFit,character'
lrTest(object, hypothesis, ...)
```

**Arguments**

- `object`: ZlmFit
- `hypothesis`: See Details
- `...`: Arguments passed on to `zlm`
  - `formula`: a formula with the measurement variable on the LHS and predictors present in colData on the RHS
  - `sca`: SingleCellAssay object
  - `method`: character vector, either 'glm', 'glmer' or 'bayesglm'
  - `silent`: Silence common problems with fitting some genes
  - `ebayes`: if TRUE, regularize variance using empirical bayes method
  - `ebayesControl`: list with parameters for empirical bayes procedure. See `ebayes`.
  - `force`: Should we continue testing genes even after many errors have occurred?
  - `hook`: a function called on the `fit` after each gene.
  - `parallel`: If TRUE and `option(mc.cores)>1` then multiple cores will be used in fitting.
  - `LMlike`: if provided, then the model defined in this object will be used, rather than following the formulas. This is intended for internal use.
  - `onlyCoef`: If TRUE then only an array of model coefficients will be returned (probably only useful for bootstrapping).
  - `exprs_values`: character or integer passed to `assay` specifying which assay to use for testing.

**Value**

3D array
Description

Methods in this package operate on log-transformed (multiplicative scale) expression. We attempt to check for this at construction, and then over-ride the `assay` method to return the “layer” containing such log-transformed data.

Usage

```r
magic_assay_names()
assay_idx(x)
```

## S4 method for signature 'SingleCellAssay,missing'
```r
assay(x, i, withDimnames = TRUE, ...)
```

Arguments

- **x** `SingleCellAssay`
- **i** must be missing for this method to apply
- **withDimnames** A logical(1), indicating whether the dimnames of the SummarizedExperiment object should be applied (i.e. copied) to the extracted assays. More precisely, setting `withDimnames=FALSE` in the getter returns the assays as-is whereas setting `withDimnames=FALSE` return them with possibly modified dimnames. Setting `withDimnames=FALSE` in the setter (`assays<-`) is required when the dimnames on the supplied assays are not identical to the dimnames on the SummarizedExperiment object; it does not influence actual assignment of dimnames to assays (they're always stored as-is).

Note that
```r
assays(x, withDimnames=FALSE) <- assays(x, withDimnames=FALSE)
```

is guaranteed to always work and be a no-op. This is not the case if `withDimnames=TRUE` is used or if `withDimnames` is not specified.

... passed to parent method

Details

By default we return the assay whose names, as given by `assayNames(x)`, matches the first element in the vector `c('thresh', 'et', 'Et', 'lCount', 'logTPM', 'logCounts', 'logcounts')`.

Functions

- `magic_assay_names()`: list of names assumed to represent log-transformed data, in order of usage preference
- `assay_idx()`: what index is returned by default by ‘assay’
Examples

```r
data(vbetaFA)
assay(vbetaFA)[1:3,1:3]
assay(vbetaFA, 'thresh', withDimnames = FALSE) = assay(vbetaFA)*0 - 9
assay(vbetaFA)[1:3, 1:3]
```

---

**MAITs data set, RNASeq**

Description

MAITs data set, RNASeq

Format

A list containing an expression matrix (`expressionmat`), cell `cdat` and feature `fdat`.

See Also

`FromMatrix`

---

**MAST-defunct**

Defunct functions in package 'MAST'

Description

These functions are defunct or have been renamed.

Functions (and replacements, if available)

- `filter` : `mast_filter`
- `cData` : `colData`
- `fData` : `mcols`
- `exprs` : `assay`
- `zlm.SingleCellAssay` : `zlm`
- `combine` : `cbind` or `rbind`
- `deviance_residuals_hook` No replacement available, underlying API changed
**mast_filter**

Filter a SingleCellAssay

**Description**
Remove, or flag wells that are outliers in discrete or continuous space.

**Usage**
mast_filter(sc, groups = NULL, filt_control = NULL, apply_filter = TRUE)

burdenOfFiltering(sc, groups, byGroup = FALSE, filt_control = NULL)

**Arguments**

- **sc** The SingleCellAssay object
- **groups** An optional character naming the grouping variable
- **filt_control** The list with configuration parameters for the filter.
- **apply_filter** logical should the filter be applied, or should a matrix of booleans giving if a well would be subject to a filtering criteria be returned?
- **byGroup** in the case of burdenOfFiltering should the filter be stratified by groups, or only the plotting.

**Details**
The function filters wells that don’t pass filtering criteria described in filt_control. filt_control is a list with named elements nOutlier (minimum number of outlier cells for a cell to be filtered [default = 2]) sigmaContinuous (the z-score outlier threshold for the continuous part of the signal) [default = 7] and sigmaProportion (the z-score outlier threshold for the discrete part of the signal) [default = 7].

If groups is provided, the filtering is calculated within each level of the group, then combined again as output.

**Value**
A filtered result

**Functions**
- burdenOfFiltering(): plot the proportions of wells are filtered due to different criteria

**Author(s)**
Andrew McDavid
See Also

burdenOfFiltering

Examples

data(vbetaFA)
## Split by 'ncells', apply to each component, then recombine
vbeta.filtered <- mast_filter(vbetaFA, groups='ncells')
## Returned as boolean matrix
was.filtered <- mast_filter(vbetaFA, apply_filter=FALSE)
## Wells filtered for being discrete outliers
head(subset(was.filtered, pctout))
burdenOfFiltering(vbetaFA, groups='ncells', byGroup=TRUE)
b burdenOfFiltering(vbetaFA, groups='ncells')

 meld_list_left

Combine lists, preferentially taking elements from x if there are duplicate names

Description

Combine lists, preferentially taking elements from x if there are duplicate names

Usage

meld_list_left(x, y)

Arguments

 x    list
 y    list

Examples

MAST:::meld_list_left(list(A=1, B=2), list(A = 0))
**melt.SingleCellAssay**

"Melt" a SingleCellAssay matrix

**Description**

Return a molten (flat) representation, taking the cross-product of the expression values, the colData (column meta data), and the feature data (mcols).

**Usage**

```r
## S3 method for class 'SingleCellAssay'
melt(data, ..., na.rm = FALSE, value.name = "value")
```

**Arguments**

- `data` SingleCellAssay
- `...` ignored
- `na.rm` ignored
- `value.name` name of 'values' column in returned value

**Value**

A data.table, with the cartesian product of the row and column attributes and the expression values

**Examples**

```r
data(vbetaFA)
melt.SingleCellAssay(vbetaFA[1:10,])
as(vbetaFA[1:10,], 'data.table')
```

---

**model.matrix**

Model matrix accessor

**Description**

Model matrix accessor

**Usage**

```r
model.matrix(object, ...)
```

```r
## S4 method for signature 'LMlike'
model.matrix(object, ...)
```
myBiplot

Arguments

object  LMlike or subclass
...  ignored

Value

model.matrix if present

Methods (by class)

- model.matrix(LMlike): return the model.matrix

Description

Replace model matrix

Usage

model.matrix(object) <- value

Arguments

object  LMlike or subclass
value  matrix

Value

modify object

myBiplot  Makes a nice BiPlot

Description

Creates a custom BiPlot for visualizing the results of PCA

Usage

myBiplot(pc, colorfactor, scaling = 100, nudge = 1.2, N = 10, dims = 1:2, ...)
new_with_repaired_slots

Arguments

- pc: output of prcomp
- colorfactor: a factor the same length as nrow(pc$x) to color the points
- scaling: integer to scale the vectors showing loadings
- nudge: numeric to offset labels for loadings
- N: number of variables with longest dim[1] or dim[2] projections to display
- dims: numeric vector of length 2 indicating which PCs to plot
- ...: passed to plot

Value

printed plot

Description

Instantiate a class, but warn rather than error for badly named slots

Usage

new_with_repaired_slots(classname, ..., extra)

Arguments

- classname: 'character' naming a class
- ...: slots in 'classname'
- extra: named list giving other slots in 'classname'

Value

'new(classname)'

Examples

MAST::new_with_repaired_slots("SimpleList", listData = list(x = LETTERS), extra = list(elementType = 'character', food = "tasty", beer = "cold"))
pbootVcov1  

*Bootstrap a zlmfit*

**Description**

Sample cells with replacement to find bootstrapped distribution of coefficients

**Usage**

```r
pbootVcov1(cl, zlmfit, R = 99)

bootVcov1(zlmfit, R = 99, boot_index = NULL)
```

**Arguments**

- **cl**: a cluster object created by `makeCluster`
- **zlmfit**: class `ZlmFit`
- **R**: number of bootstrap replicates
- **boot_index**: list of indices to resample. Only one of R or boot_index can be offered.

**Value**

array of bootstrapped coefficients

**Functions**

- `pbootVcov1()`: parallel version of bootstrapping

**Examples**

```r
data(vbetaFA)
zlmVbeta <- zlm(~ Stim.Condition, subset(vbetaFA, ncells==1)[1:5,])
#Only run 3 bootstrap, which you wouldn’t ever want to do in practice...
bootVcov1(zlmVbeta, R=3)
```
plot.thresholdSCRNACountMatrix

Plot cutpoints and densities for thresholding

Description

Plot cutpoints and densities for thresholding

Usage

```r
## S3 method for class 'thresholdSCRNACountMatrix'
plot(x, ask = FALSE, wait.time = 0, type = "bin", indices = NULL, ...)
```

Arguments

- **x**: output of `thresholdSCRNACountMatrix`
- **ask**: if TRUE then will prompt before displaying each plot
- **wait.time**: pause (in seconds) between each plot
- **type**: one or more of the following: 'bin' (plot the genes by the binning used for thresholding), or 'gene' (plot thresholding by gene – see next argument)
- **indices**: if type is equal to 'gene', and is a integer of length 1, then a random sample of indices genes is taken. If it is NULL, then 10 genes are sampled. If it is a integer vector of length > 1, then it is interpreted as giving a list of indices of genes to be displayed.
- **...**: further arguments passed to `plot`

Value

- displays plots

Examples

```r
## See thresholdSCRNACountMatrix
example(thresholdSCRNACountMatrix)
```

plotlrt

Plot a likelihood ratio test object

Description

Constructs a forest-like plot of signed log10 p-values, possibly adjusted for multiple comparisons. adjust can be one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".
Usage

plotlrt(lr, adjust = "fdr", thres = 0.1, trunc = 1e-06, groups = NULL)

Arguments

lr output from lrtest, with returnall=FALSE
adjust character, passed along to p.adjust, see below
thres numeric genes with adjusted pvalues above this value are not depicted
trunc numeric p values below this value are truncated at this value
groups character grouping value. If provided, must match groups argument passed to lrtest. Plots done separately for each group.

Value

Constructs a dotplot

Author(s)

andrew

plotSCAConcordance Concordance plots of filtered single vs n-cell assays

Description

Plot the average expression value of two subsets of the data. Generally these might be 1 cell and multiple-cell replicates, in which case if the ncol column ncells is set then the averages will be adjusted accordingly. But it could be any grouping.

Usage

plotSCAConcordance(
  SCellAssay,
  NCellAssay,
  filterCriteria = list(nOutlier = 2, sigmaContinuous = 9, sigmaProportion = 9),
  groups = NULL,
  ...
)

Arguments

SCellAssay is a FluidigmAssay for the 1-cell per well assay
NCellAssay is a FluidigmAssay for the n-cell per well assay
filterCriteria is a list of filtering criteria to apply to the SCellAssay and NCellAssay
groups is a character vector naming the group within which to perform filtering. NULL by default.
... passed to getConcordance
predict.ZlmFit

Value

printed plot

See Also

getConcordance

Examples

data(vbetaFA)
sca1 <- subset(vbetaFA, ncells==1)
sca100 <- subset(vbetaFA, ncells==100)
plotSCAConcordance(sca1, sca100)

predict.ZlmFit Return predictions from a ZlmFit object.

Description

Return predictions from a ZlmFit object.

Usage

## S3 method for class 'ZlmFit'
predict(object, newdata = NULL, modelmatrix = NULL, ...)

Arguments

object A ZlmFit

newdata The data to predict from. Currently ignored, will use the data in the object.

modelmatrix The model matrix specifying the linear combination of coefficients.

... ignored

Value

Predictions (on the link scale) and standard errors.

Examples

##See stat_ell
example(stat_ell)
predicted_sig  Predicted signatures

Description
Predicted signatures

Format
A data frame of predicted gene expression signatures for stimulated and unstimulated cells.

primerAverage  Average expression values for duplicated/redundant genes

Description
Takes an average, potentially on a different scale given by fun.natural of some genes. The average is then transformed with fun.cycle.

Usage
primerAverage(fd, geneGroups, fun.natural = expavg, fun.cycle = logshift)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fd</td>
<td>SingleCellAssay or subclass</td>
</tr>
<tr>
<td>geneGroups</td>
<td>character naming a column in the featureData that keys the duplicates</td>
</tr>
<tr>
<td>fun.natural</td>
<td>transformation to be used to collapse the duplicate expression values</td>
</tr>
<tr>
<td>fun.cycle</td>
<td>transformation to be used after collapsing</td>
</tr>
</tbody>
</table>

Value
averaged version of fd.

Note
This code needs to be tested more extensively after a refactoring. Caveat calculator.
print.summaryZlmFit  

Print summary of a ZlmFit

Description

Shows the top ‘n’ genes by z score on ‘by’

Usage

```r
## S3 method for class 'summaryZlmFit'
print(x, n = 2, by = "logFC", ...)
```

Arguments

- `x` output from `summary(ZlmFit)`
- `n` number of genes to show
- `by` one of ‘C’, ‘D’ or ‘logFC’ for continuous, discrete and log fold change z-scores for each contrast
- `...` ignored

Value

prints a pretty table and invisibly returns a `data.table` representing the table.

See Also

`summary.ZlmFit-method`

read.fluidigm  

Reads a Fluidigm Biomark (c. 2011) raw data file (or set of files)

Description

This function reads a raw Fluidigm Biomark data file or set of files and constructs a `SingleCellAssay` (or `FluidigmAssay`) object. This was written c. 2011 and has not been tested lately. The Biomark format may have changed.
Usage

```r
read.fluidigm(
  files = NULL,
  metadata = NULL,
  header.size = 2,
  skip = 8,
  cycle.threshold = 40,
  metadataColClasses = NULL,
  meta.key = NULL,
  idvars = NULL,
  splitby = NULL,
  unique.well.id = "Chamber.ID",
  raw = TRUE,
  assay = NULL,
  geneid = "Assay.Name",
  sample = NULL,
  well = "Well",
  measurement = "X40.Ct",
  measurement.processed = "Ct",
  ncells = "SampleRConc"
)
```

Arguments

- **files**: A character vector of files to read.
- **metadata**: A character path and filename of a CSV file containing additional metadata about the samples.
- **header.size**: A numeric indicating the number of lines in the header (default 2).
- **skip**: Numeric how many lines to skip before reading (default 8).
- **cycle.threshold**: The maximum number of PCR cycles performed (default 40) numeric.
- **metadataColClasses**: Optional character vector giving the column classes of the metadata file. See `read.table`.
- **meta.key**: Optional character vector that identifies the key column between the metadata and the fluidigm data.
- **idvars**: Optional character vector that defines the set of columns uniquely identifying a well (unique cell, gene, and condition).
- **splitby**: Optional character that defines the column / variable used to split the resulting data into a list of SingleCellAssay, such that unique levels of `splitby` each fall into their own SingleCellAssay. Usually the experimental unit subjected to different treatments.
- **unique.well.id**: The column that uniquely identifies a sample well in the data. Default is "Chamber.ID".
- **raw**: Logical flag indicating this is raw data coming off the instrument. Thus we make some assumptions about the column names that are present.
**assay** character name of a column that uniquely identifies an Assay (i.e. gene). Default is NULL.

**geneid** character names of the column that identifies a gene. Default is "Assay.Name"

**sample** character name of a column that uniquely identifies a sample

**well** character name of a column that uniquely identifies a well. Default "Well".

**measurement** character name of the column that holds the measurement. Default "X40.Ct".

**measurement.processed** character one of "Ct","40-Ct", or "et". If not "Ct", the measurement will be transformed.

**ncells** The column with the number of cells in this well.

**Value**

list of SingleCellAssay holding the data.

**Author(s)**

Greg Finak

---

**Description**

The order of terms will be rearrange to suit R’s liking for hierarchy but otherwise the function should be idempotent for

**Usage**

```r
removeResponse(Formula, warn = TRUE)
```

**Arguments**

- **Formula** formula
- **warn** Issue a warning if a response variable is found?

**Value**

formula

**Author(s)**

Andrew
### rstandard.bayesglm

*rstandard for bayesglm objects.*

**Description**

rstandard bayesglm object S3 method

**Usage**

```r
## S3 method for class 'bayesglm'
rstandard(
  model,
  infl = influence(model, do.coef = FALSE),
  type = c("deviance", "pearson"),
  ...,
)
```

**Arguments**

- `model` bayesglm
- `infl` see `rstandard`
- `type` see `rstandard`
- `...` ignored

**Value**

numeric residuals

---

### SceToSingleCellAssay

*Coerce a SingleCellExperiment to some class defined in MAST*

**Description**

Coerce a SingleCellExperiment to some class defined in MAST

**Usage**

```r
SceToSingleCellAssay(sce, class = "SingleCellAssay", check_sanity = TRUE)
```

**Arguments**

- `sce` object inheriting from SingleCellExperiment
- `class` character naming the class to be coerced to
- `check_sanity` (default: TRUE) Set FALSE to override sanity checks that try to ensure that the default assay is log-transformed and has at least one exact zero. See defaultAssay for details on the "default assay" which is assumed to contain log transformed data.
**se.coef**  

**Value**  
object of the indicated class.

**Description**  
Given a fitted model, return the standard errors of the coefficient

**Usage**  
se.coef(object, ...)

**Arguments**  

object  
a model implementing vcov

...  
passed to methods

**Value**  
vector or matrix

**See Also**  
ZlmFit-class

**Examples**  
#see ZlmFit-class for examples  
example('ZlmFit-class')

**Description**  
Display info

**Usage**  

## S4 method for signature 'LMlike'  
show(object)

## S4 method for signature 'ZlmFit'  
show(object)
split,SingleCellAssay,character-method

Arguments

object an object of some type

Details

Prints information on a LMlike object

Value

side effect of printing to console

Methods (by class)

• show(ZlmFit): print info on ZlmFit

split,SingleCellAssay,character-method

Split into list

Description

Splits a SingleCellAssay into a list by a factor (or something coercible into a factor) or a character giving a column of colData(x)

Usage

## S4 method for signature 'SingleCellAssay,character'
split(x, f, drop = FALSE, ...)

Arguments

x SingleCellAssay
f length-1 character, or atomic of length ncol(x)
drop drop unused factor levels
... ignored

Value

List

Examples

data(vbetaFA)
split(vbetaFA, 'ncells')
fa <- as.factor(colData(vbetaFA)$ncells)
split(vbetaFA, fa)
stat_ell  

Plot confidence ellipse in 2D

Description

The focus of the ellipse will be the point \((x, y)\) and semi-major axes aligned with the coordinate axes and scaled by \(xse, yse\) and the level.

Usage

```r
stat_ell(
  mapping = NULL,
  data = NULL,
  geom = "polygon",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  fill = NA,
  level = 0.95,
  lty = 2,
  invert = FALSE,
  alpha = 1,
  ...)
```

Arguments

- **mapping** 
  Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

- **data** 
  The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot`. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify` for which variables will be created.

- **geom** 
  The geometric object to use display the data

- **position** 
  Position adjustment, either as a string, or the result of a call to a position adjustment function.

- **na.rm** 
  If `FALSE` (the default), removes missing values with a warning. If `TRUE` silently removes missing values.

- **show.legend** 
  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
**subset,SingleCellAssay-method**

**Description**

Evaluates the expression in ... in the context of `colData(x)` and returns a subsetted version of `x`.

**Usage**

```r
## S4 method for signature 'SingleCellAssay'
subset(x, ...)```

**Examples**

```r
data(vbetaFA)
library(ggplot2)

zlmCond <- zlm(~Stim.Condition, vbetaFA[1:10,])
MM <- model.matrix(~Stim.Condition, unique(colData(vbetaFA)[,c("Stim.Condition"),drop=FALSE]))
predicted <- predict(zlmCond,modelmatrix=MM)
plt <- ggplot(predicted)+aes(x=invlogit(etaD),y=muC,xse=seD,yse=seC,col=sample)+
  facet_wrap(~primerid,scales="free_y")+theme_linedraw()+
  geom_point(size=0.5)+scale_x_continuous("Proportion expression")+
  scale_y_continuous("Estimated Mean")+
  stat_ell(aes(x=etaD,y=muC),level=0.95, invert='x')

## plot with inverse logit transformed x-axis
print(plt)

# doesn't do anything in this case because there are no inestimable coefficients
predictI <- impute(predicted, groupby='primerid')
```
Arguments

  x   SingleCellAssay

  ... expression

Value

  SingleCellAssay

Examples

  data(vbetaFA)
  subset(vbetaFA, ncells==1)

summarize  Return programmatically useful summary of a fit

Description

  Return programmatically useful summary of a fit

Usage

  summarize(object, ...)

Arguments

  object   LMlike or subclass

  ... other arguments

Value

  list of parameters characterizing fit
Summary,GSEATests-method

Summarize gene set enrichment tests

Description

Returns a data.table with one row per gene set. This data.table contains columns:

- **set**: name of gene set
- **cond_Z**: Z statistic for continuous component
- **cont_P**: wald P value
- **cont_effect**: difference in continuous regression coefficients between null and test sets (i.e., the numerator of the Z-statistic.)
- **disc_Z**: Z statistic for discrete
- **disc_P**: wald P value
- **disc_effect**: difference in discrete regression coefficients between null and test sets.
- **combined_Z**: combined discrete and continuous Z statistic using Stouffer’s method
- **combined_P**: combined P value
- **combined_adj**: FDR adjusted combined P value

Usage

```r
## S4 method for signature 'GSEATests'
summary(object, ...)
```

Arguments

- **object**: A GSEATests object
- **...**: passed to calcZ

Value

data.table

See Also

gseaAfterBoot

Examples

```r
## See the examples in gseaAfterBoot
eexample(gseaAfterBoot)
```
**summary.ZlmFit-method**  
*Summarize model features from a ZlmFit object*

**Description**

Returns a data.table with a special print method that shows the top 2 most significant genes by contrast. This data.table contains columns:

- **primerid** the gene
- **component** C=continuous, D=discrete, logFC=log fold change, S=combined using Stouffer's method, H=combined using hurdle method
- **contrast** the coefficient/contrast of interest
- **ci.hi** upper bound of confidence interval
- **ci.lo** lower bound of confidence interval
- **coef** point estimate
- **z** z score (coefficient divided by standard error of coefficient)
- **Pr(>Chisq)** likelihood ratio test p-value (only if doLRT=TRUE)

Some of these columns will contain NAs if they are not applicable for a particular component or contrast.

**Usage**

```r
## S4 method for signature 'ZlmFit'
summary(
  object,
  logFC = TRUE,
  doLRT = FALSE,
  level = 0.95,
  parallel = FALSE,
  ...)
```

**Arguments**

- **object** A ZlmFit object
- **logFC** If TRUE, calculate log-fold changes, or output from a call to getLogFC.
- **doLRT** if TRUE, calculate lrTests on each coefficient, or a character vector of such coefficients to consider.
- **level** what level of confidence coefficient to return. Defaults to 95 percent.
- **parallel** If TRUE and option(mc.cores)>1 then multiple cores will be used in fitting.
- ... ignored
Summary of the effect of thresholding

Description

Returns the proportion of (putative) expression, the variance of expressed cells, and -log10 shapiro-wilk tests for normality on the expressed cells

Usage

## S3 method for class 'thresholdSCRNAcountMatrix'
summary(object, ...)

## S3 method for class 'summaryThresholdSCRNA'
print(x, ...)

Arguments

object    a thresholdSCRNAcountMatrix
...
          currently ignored
x          a summaryThresholdSCRNA object, ie output from summary.thresholdSCRNAcountMatrix

Value

a list of statistics on the original data, and thresholded data
thresholdSCRNACountMatrix

Threshold a count matrix using an adaptive threshold.

Description
An adaptive threshold is calculated from the conditional mean of expression, based on 10 bins of the genes with similar expression levels. Thresholds are chosen by estimating cutpoints in the bimodal density estimates of the binned data. These density estimates currently exclude the zeros due to complications with how the bandwidth is selected. (If the bandwidth is too small, then extra peaks/modes are found and everything goes haywire). If the diagnostic plots don’t reveal any bimodal bins, this is probably the reason, and you may not need to threshold since background in the data are exact zeros.

Usage

thresholdSCRNACountMatrix(
  data_all,
  conditions = NULL,
  cutbins = NULL,
  nbins = 10,
  bin_by = "median",
  qt = 0.975,
  min_per_bin = 50,
  absolute_min = 0,
  data_log = TRUE,
  adj = 1
)

Arguments

data_all matrix of (possibly log-transformed) counts or TPM. Rows are genes and columns are cells.
conditions Bins are be determined per gene and per condition. Typically contrasts of interest should be specified.
cutbins vector of cut points.
nbins integer number of bins when cutbins is not specified.
bin_by character "median", "proportion", "mean"
qt when bin_by is "quantile", what quantile should be used to form the bins
min_per_bin minimum number of genes within a bin
absolute_min  numeric giving a hard threshold below which everything is assumed to be noise
data_log      is data_all log+1 transformed? If so, it will be returned on the (log+1)-scale as well.
adj           bandwith adjustment, passed to density

Value

list of thresholded counts (on natural scale), thresholds, bins, densities estimated on each bin, and the original data

Examples

data(maits, package='MAST', envir = environment())
sca <- FromMatrix(t(maits$expressionmat[,1:1000]), maits$cdat, maits$fdat[1:1000,])
tt <- thresholdSCRNACountMatrix(assay(sca))
tt <- thresholdSCRNACountMatrix(2^assay(sca)-1, data_log=FALSE)
opar <- par(no.readonly = TRUE)
on.exit(par(opar))
par(mfrow=c(4,2))
plot(tt)

vbeta

Vbeta Data Set

Description

Vbeta Data Set

Format

a data frame with 11 columns. Column Ct contains the cycle threshold, with NA denoting that the threshold was never crossed. So it is inversely proportional to the log2 mRNA, and should be negated (and NAs set to zero) if it is used as a expression measurement for a FluidigmAssay.

vbetaFA

Vbeta Data Set, FluidigmAssay

Description

Vbeta Data Set, FluidigmAssay

Format

a FluidigmAssay of the vbeta data set.

See Also

vbeta, FromFlatDF
waldTest

Run a Wald test

Description

Run a Wald tests on discrete and continuous components hypothesis can be one of a character giving complete factors or terms to be dropped from the model, CoefficientHypothesis giving names of coefficients to be dropped, Hypothesis giving contrasts using the symbolically, or a contrast matrix, with one row for each coefficient in the full model, and one column for each contrast being tested.

Usage

waldTest(object, hypothesis)

Arguments

object LMlike or subclass
hypothesis the hypothesis to be tested. See details.

Value

array giving test statistics

See Also

fit
lrTest
lht

Examples

#see ZlmFit-class for examples
example('ZlmFit-class')

waldTest,ZlmFit,matrix-method

Wald test

Description

A 3D array with first dimension being the genes, next dimension giving information about the test (the degrees of freedom, Chisq statistic, and P value), and final dimension being the value of these quantities on the discrete, continuous and hurdle (combined) levels.
Usage

```r
## S4 method for signature 'ZlmFit,matrix'
waldTest(object, hypothesis)
```

Arguments

- `object`: ZlmFit
- `hypothesis`: See Details

Value

3D array

---

```r
xform(mat, scale = FALSE)
```

Arguments

- `mat`: matrix (such as produced by exprs)
- `scale`: should the columns also be scaled to have unit variance

Value

matrix

---

**zlm**

Zero-inflated regression for SingleCellAssay

Description

For each gene in sca, fits the hurdle model in formula (linear for et>0), logistic for et==0 vs et>0. Return an object of class ZlmFit containing slots giving the coefficients, variance-covariance matrices, etc. After each gene, optionally run the function on the fit named by 'hook'

---

**xform**

Make matrix of continuous expression values, orthogonal to discrete

Description

This centers each column of mat around the mean of its non-zero values.
Usage

\[ zlm(\) \]

Arguments

\begin{itemize}
\item **formula** a formula with the measurement variable on the LHS and predictors present in \texttt{colData} on the RHS
\item **sca** SingleCellAssay object
\item **method** character vector, either ‘glm’, ‘glmer’ or ‘bayesglm’
\item **silent** Silence common problems with fitting some genes
\item **ebayes** if TRUE, regularize variance using empirical bayes method
\item **ebayesControl** list with parameters for empirical bayes procedure. See \texttt{ebayes}.
\item **force** Should we continue testing genes even after many errors have occurred?
\item **hook** a function called on the \texttt{fit} after each gene.
\item **parallel** If TRUE and \texttt{option(mc.cores)>1} then multiple cores will be used in fitting.
\item **LMlike** if provided, then the model defined in this object will be used, rather than following the formulas. This is intended for internal use.
\item **onlyCoef** If TRUE then only an array of model coefficients will be returned (probably only useful for bootstrapping).
\item **exprs_values** character or integer passed to ‘assay’ specifying which assay to use for testing
\item \ldots arguments passed to the S4 model object upon construction. For example, \texttt{fitArgsC} and \texttt{fitArgsD}, or \texttt{coefPrior}.
\end{itemize}

Value

a object of class \texttt{ZlmFit} with methods to extract coefficients, etc. OR, if data is a \texttt{data.frame} just a list of the discrete and continuous fits.
Empirical Bayes variance regularization

The empirical bayes regularization of the gene variance assumes that the precision (1/variance) is drawn from a gamma distribution with unknown parameters. These parameters are estimated by considering the distribution of sample variances over all genes. The procedure used for this is determined from `ebayesControl`, a named list with components 'method' (one of 'MOM' or 'MLE') and 'model' (one of 'H0' or 'H1') method MOM uses a method-of-moments estimator, while MLE using the marginal likelihood. H0 model estimates the precisions using the intercept alone in each gene, while H1 fits the full model specified by formula

See Also

ZlmFit-class, ebayes, GLMlike-class, BayesGLMlike-class

Examples

```r
data(vbetaFA)
zlmVbeta <- zlm(~ Stim.Condition, subset(vbetaFA, ncells==1)[1:10,])
slotNames(zlmVbeta)
#A matrix of coefficients
coe(zlmVbeta, 'D')['CCL2']
#An array of covariance matrices
vcov(zlmVbeta, 'D')[,,'CCL2']
waldTest(zlmVbeta, CoefficientHypothesis('Stim.ConditionUnstim'))
```

## Can also provide just a `data.frame` instead
data<- data.frame(x=rnorm(500), z=rbinom(500, 1, .3))
logit.y <- with(data, x*x2 + z*x2); mu.y <- with(data, 10+10*x+10*z + rnorm(500))
y <- (runif(500)<exp(logit.y)/(1+exp(logit.y)))*1
y[y>0] <- mu.y[y>0]
data$y <- y
fit <- zlm(y ~ x+z, data)
summary.glm(fit$disc)
```
## S4 method for signature 'ZlmFit,matrix'
lrTest(object, hypothesis, ...)

## S4 method for signature 'ZlmFit,CoefficientHypothesis'
waldTest(object, hypothesis)

## S4 method for signature 'ZlmFit,Hypothesis'
waldTest(object, hypothesis)

## S4 method for signature 'ZlmFit'
coef(object, which, ...)

## S4 method for signature 'ZlmFit'
vcov(object, which, ...)

## S4 method for signature 'ZlmFit'
se.coef(object, which, ...)

### Arguments

- **object**: ZlmFit
- **hypothesis**: call to **Hypothesis** or **CoefficientHypothesis** or a matrix giving such contrasts.
- **...**: ignored
- **which**: character vector, one of "C" (continuous) or "D" (discrete) specifying which component should be returned

### Value

see "Methods (by generic)"

### Methods (by generic)

- **lrTest(object = ZlmFit, hypothesis = CoefficientHypothesis)**: Returns an array with likelihood-ratio tests on contrasts defined using **CoefficientHypothesis**().
- **lrTest(object = ZlmFit, hypothesis = Hypothesis)**: Returns an array with likelihood-ratio tests specified by **Hypothesis**, which is a **Hypothesis**.
- **lrTest(object = ZlmFit, hypothesis = matrix)**: Returns an array with likelihood-ratio tests specified by **Hypothesis**, which is a contrast matrix.
- **waldTest(object = ZlmFit, hypothesis = CoefficientHypothesis)**: Returns an array with Wald Tests on contrasts defined using **CoefficientHypothesis**().
- **waldTest(object = ZlmFit, hypothesis = Hypothesis)**: Returns an array with Wald Tests on contrasts defined in **Hypothesis**()
- **coef(ZlmFit)**: Returns the matrix of coefficients for component **which**.
- **vcov(ZlmFit)**: Returns an array of variance/covariance matrices for component **which**.
- **se.coef(ZlmFit)**: Returns a matrix of standard error estimates for coefficients on component **which**.
ZlmFit-class

Slots

coeffC  matrix of continuous coefficients  
coeffD  matrix of discrete coefficients  
vcovc  array of variance/covariance matrices for coefficients  
vcovd  array of variance/covariance matrices for coefficients  
LMlike  the LmWrapper object used  
sca  the SingleCellAssay object used  
deviance  matrix of deviances  
loglik  matrix of loglikelihoods  
df.null  matrix of null (intercept only) degrees of freedom  
df.resid  matrix of residual DOF  
dispersion  matrix of dispersions (after shrinkage)  
dispersionNoShrink  matrix of dispersion (before shrinkage)  
priorDOF  shrinkage weight in terms of number of psuedo-obs  
priorVar  shrinkage target  
converged  output that may optionally be set by the underlying modeling function  
hookOut  a list of length ngenes containing output from a hook function, if zlm was called with one exprs_values 'character' or 'integer' with the 'assay' used.

See Also

zlm summary, ZlmFit-method

Examples

data(vbetaFA)
zlmVbeta <- zlm(~ Stim.Condition+Population, subset(vbetaFA, ncells==1)[1:10,])
#Coefficients and standard errors
coef(zlmVbeta, 'D')
coef(zlmVbeta, 'C')
se.coef(zlmVbeta, 'C')
#Test for a Population effect by dropping the whole term (a 5 degree of freedom test)
lrTest(zlmVbeta, 'Population')
#Test only if the VbetaResponsive cells differ from the baseline group
lrTest(zlmVbeta, CoefficientHypothesis('PopulationVbetaResponsive'))
# Test if there is a difference between CD154+/Unresponsive and CD154-/Unresponsive.
# Note that because we parse the expression
# the columns must be enclosed in backquotes
# to protect the \quote{+} and \quote{-} characters.
lrTest(zlmVbeta, Hypothesis(`PopulationCD154+VbetaUnresponsive` - `PopulationCD154-VbetaUnresponsive`))
waldTest(zlmVbeta, Hypothesis(`PopulationCD154+VbetaUnresponsive` - `PopulationCD154-VbetaUnresponsive`))
Index

applyFlat, 4
assay, 36
assay, SingleCellAssay, missing-method
  (magic_assay_names), 35
assay_idx (magic_assay_names), 35
BayesGLMlike-class, 5
bootVcov1 (pbootVcov1), 42
burdenOfFiltering (mast_filter), 37
calcZ, 5, 22
cbind, 36
cData (MAST-defunct), 36
coef, LMERlike-method (LMERlike-class), 26
coef, ZlmFit-method (ZlmFit-class), 64
CoefficientHypothesis (Hypothesis), 24
colData, 36
colData<-, SingleCellAssay, DataFrame-method, 6
collectResiduals, 7
combined_residuals_hook
  (collectResiduals), 7
computeEtFromCt, 8
condmean (freq), 15
condSd (freq), 15
continuous_residuals_hook
  (collectResiduals), 7
convertMASTClassicToSingleCellAssay, 9
CovFromBoots, 10
defaultAssay, 16, 18, 50
defaultAssay (magic_assay_names), 35
defaultPrior, 11
deviance_residuals_hook
  (collectResiduals), 7
discrete_residuals_hook
  (collectResiduals), 7
dof, 11
dof, GLMlike-method (dof), 11
dof, LMERlike-method (dof), 11
Drop, 12
ebayes, 12, 34, 63
expavg, 13, 31
expres (MAST-defunct), 36
fData (MAST-defunct), 36
filter (MAST-defunct), 36
filterLowExpressedGenes, 14
fit, 14
fit, BayesGLMlike, missing-method (fit), 14
fit, GLMlike, missing-method (fit), 14
fit, LMERlike, missing-method (fit), 14
fitted_phat (collectResiduals), 7
FluidigmAssay, 16
FluidigmAssay (FromFlatDF), 16
freq, 15
FromFlatDF, 16, 60
FromMatrix, 17, 36
getConcordance, 18
getLogFC (logFC), 30
getrc (getConcordance), 18
getss (getConcordance), 18
getwellKey, 20
getwellKey, SingleCellAssay-method
  (getwellKey), 20
getwss (getConcordance), 18
GLMlike-class, 20
gsea_control (gseaAfterBoot), 21
gseaAfterBoot, 21
GSEATests-class, 23
hushWarning, 23
Hypothesis, 24, 30, 31, 65
impute, 24
influence.bayesglm, 25
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>influence.glm</td>
<td>25</td>
</tr>
<tr>
<td>invlogit</td>
<td>26</td>
</tr>
<tr>
<td>LMERlike-class</td>
<td>26</td>
</tr>
<tr>
<td>LMLike-class</td>
<td>27</td>
</tr>
<tr>
<td>logFC</td>
<td>30</td>
</tr>
<tr>
<td>logLik, GLMLike-method (LMLike-class)</td>
<td>27</td>
</tr>
<tr>
<td>logLik, LMERlike-method (LMERlike-class)</td>
<td>26</td>
</tr>
<tr>
<td>logmean</td>
<td>31</td>
</tr>
<tr>
<td>LRT</td>
<td>32</td>
</tr>
<tr>
<td>LRT, SingleCellAssay, character-method (LRT)</td>
<td>32, 33</td>
</tr>
<tr>
<td>lrTest, LMLike, character-method (LMLike-class)</td>
<td>27</td>
</tr>
<tr>
<td>lrTest, LMLike, CoefficientHypothesis-method (LMLike-class)</td>
<td>27</td>
</tr>
<tr>
<td>lrTest, LMLike, Hypothesis-method (LMLike-class)</td>
<td>27</td>
</tr>
<tr>
<td>lrTest, LMLike, matrix-method (LMLike-class)</td>
<td>27</td>
</tr>
<tr>
<td>lrTest, ZlmFit, character-method (ZlmFit-class)</td>
<td>34</td>
</tr>
<tr>
<td>lrTest, ZlmFit, CoefficientHypothesis-method (ZlmFit-class)</td>
<td>64</td>
</tr>
<tr>
<td>lrTest, ZlmFit, Hypothesis-method (ZlmFit-class)</td>
<td>64</td>
</tr>
<tr>
<td>lrTest, ZlmFit, matrix-method (ZlmFit-class)</td>
<td>64</td>
</tr>
<tr>
<td>magic_assay_names</td>
<td>35</td>
</tr>
<tr>
<td>maits</td>
<td>36</td>
</tr>
<tr>
<td>MAST (MAST-package)</td>
<td>4</td>
</tr>
<tr>
<td>MAST-defunct</td>
<td>36</td>
</tr>
<tr>
<td>MAST-package</td>
<td>4</td>
</tr>
<tr>
<td>mast_filter</td>
<td>37</td>
</tr>
<tr>
<td>mcols</td>
<td>36</td>
</tr>
<tr>
<td>meld_list_left</td>
<td>38</td>
</tr>
<tr>
<td>melt.SingleCellAssay</td>
<td>39</td>
</tr>
<tr>
<td>model.matrix</td>
<td>39</td>
</tr>
<tr>
<td>model.matrix, LMLike-method (model.matrix)</td>
<td>39</td>
</tr>
<tr>
<td>model.matrix&lt;-</td>
<td>40</td>
</tr>
<tr>
<td>myBiplot</td>
<td>40</td>
</tr>
<tr>
<td>new_with_repaired_slots</td>
<td>41</td>
</tr>
<tr>
<td>numexp (freq)</td>
<td>15</td>
</tr>
<tr>
<td>partialScore (collectResiduals)</td>
<td>7</td>
</tr>
<tr>
<td>pbootVcov1</td>
<td>42</td>
</tr>
<tr>
<td>plot.thresholdSCRNACountMatrix</td>
<td>43</td>
</tr>
<tr>
<td>plotlrt</td>
<td>43</td>
</tr>
<tr>
<td>plotSCAConcordance</td>
<td>19, 44</td>
</tr>
<tr>
<td>predict.ZlmFit</td>
<td>45</td>
</tr>
<tr>
<td>predicted_sig</td>
<td>46</td>
</tr>
<tr>
<td>primerAverage</td>
<td>46</td>
</tr>
<tr>
<td>print.summaryThresholdSCRNA</td>
<td>58</td>
</tr>
<tr>
<td>print.summaryZlmFit</td>
<td>47</td>
</tr>
<tr>
<td>rbind</td>
<td>36</td>
</tr>
<tr>
<td>read.fluidigm</td>
<td>47</td>
</tr>
<tr>
<td>read.table</td>
<td>48</td>
</tr>
<tr>
<td>removeResponse</td>
<td>49</td>
</tr>
<tr>
<td>rstandard</td>
<td>50</td>
</tr>
<tr>
<td>rstandard.bayesglm</td>
<td>50</td>
</tr>
<tr>
<td>SceToSingleCellAssay</td>
<td>50</td>
</tr>
<tr>
<td>se.coef</td>
<td>51</td>
</tr>
<tr>
<td>se.coef, ZlmFit-method (ZlmFit-class)</td>
<td>64</td>
</tr>
<tr>
<td>show, LMLike-method</td>
<td>51</td>
</tr>
<tr>
<td>show, ZlmFit-method</td>
<td>51</td>
</tr>
<tr>
<td>SingleCellAssay (FromFlatDF)</td>
<td>16</td>
</tr>
<tr>
<td>split, SingleCellAssay, character-method</td>
<td>52</td>
</tr>
<tr>
<td>split, SingleCellAssay, factor-method (split, SingleCellAssay, character-method)</td>
<td>52</td>
</tr>
<tr>
<td>split, SingleCellAssay, list-method (split, SingleCellAssay, character-method)</td>
<td>52</td>
</tr>
<tr>
<td>stat_ell</td>
<td>53</td>
</tr>
<tr>
<td>subset, SingleCellAssay-method</td>
<td>54</td>
</tr>
<tr>
<td>summarize</td>
<td>55</td>
</tr>
<tr>
<td>summary, GSEATests-method</td>
<td>56</td>
</tr>
<tr>
<td>summary, LMLike-method (LMLike-class)</td>
<td>27</td>
</tr>
<tr>
<td>summary, ZlmFit-method (summary, ZlmFit-method)</td>
<td>57</td>
</tr>
<tr>
<td>summary, thresholdSCRNACountMatrix</td>
<td>58</td>
</tr>
<tr>
<td>thresholdSCRNACountMatrix</td>
<td>59</td>
</tr>
<tr>
<td>update, LMERlike-method (LMERlike-class)</td>
<td>26</td>
</tr>
<tr>
<td>update, LMLike-method (LMLike-class)</td>
<td>27</td>
</tr>
</tbody>
</table>
INDEX

vbeta, 60, 60
vbetafa, 60
vcov, GLMlike-method (GLMlike-class), 20
vcov, LMERlike-method (LMERlike-class), 26
vcov, ZlmFit-method (ZlmFit-class), 64
waldTest, 61
waldTest, LMLike, CoefficientHypothesis-method (LMLike-class), 27
waldTest, LMLike, matrix-method (LMLike-class), 27
waldTest, ZlmFit, CoefficientHypothesis-method (ZlmFit-class), 64
waldTest, ZlmFit, Hypothesis-method (ZlmFit-class), 64
waldTest, ZlmFit, matrix-method, 61
xform, 62
zlm, 34, 36, 62
ZlmFit (ZlmFit-class), 64
ZlmFit-class, 64
ZlmFit-summary, (summary, ZlmFit-method, 57