Package ‘MAST’

May 4, 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)
Collate 'AllGenerics.R' 'AllClasses.R' 'CovFromBoots.R'
   'Fluidigm-methods.R' 'GSEA-by-boot.R' 'Hypothesis.R'
   'LmWrapper.R' 'MAST-package.R' 'MultidimensionalScaling.R'
   'UtilityFunctions.R' 'ZlmFit-bootstrap.R' 'ZlmFit-logFC.R'
   'ZlmFit.R' 'bayesglm.R' 'convertMASTclassic.R'
   'ebayes-helpers.R' 'filterEval.R' 'helper-methods.R'
   'lmWrapper-bayesglm.R' 'lmWrapper-glm.R' 'lmWrapper-glmer.R'
   'lmWrapper-ridge.R' 'lrtest.R' 'predict.R' 'stat_ell.R'
   'thresholdSCRNA.R' 'zeroinf.R' 'zlmHooks.R'
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: 2024-05-03

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

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

Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAST-package</td>
<td>4</td>
</tr>
<tr>
<td>applyFlat</td>
<td>4</td>
</tr>
<tr>
<td>BayesGLMLike-class</td>
<td>5</td>
</tr>
<tr>
<td>calcZ</td>
<td>5</td>
</tr>
<tr>
<td>colData&lt;-SingleCellAssay,DataFrame-method</td>
<td>6</td>
</tr>
<tr>
<td>collectResiduals</td>
<td>7</td>
</tr>
<tr>
<td>computeEtFromCt</td>
<td>8</td>
</tr>
<tr>
<td>convertMASTClassicToSingleCellAssay</td>
<td>9</td>
</tr>
<tr>
<td>CovFromBoots</td>
<td>10</td>
</tr>
<tr>
<td>defaultPrior</td>
<td>11</td>
</tr>
<tr>
<td>dof</td>
<td>11</td>
</tr>
<tr>
<td>Drop</td>
<td>12</td>
</tr>
<tr>
<td>ebayes</td>
<td>12</td>
</tr>
<tr>
<td>expavg</td>
<td>13</td>
</tr>
<tr>
<td>filterLowExpressedGenes</td>
<td>14</td>
</tr>
<tr>
<td>fit</td>
<td>14</td>
</tr>
<tr>
<td>freq</td>
<td>15</td>
</tr>
<tr>
<td>FromFlatDF</td>
<td>16</td>
</tr>
<tr>
<td>FromMatrix</td>
<td>17</td>
</tr>
<tr>
<td>getConcordance</td>
<td>18</td>
</tr>
<tr>
<td>getwellKey</td>
<td>20</td>
</tr>
<tr>
<td>GLMLike-class</td>
<td>20</td>
</tr>
<tr>
<td>gseaAfterBoot</td>
<td>21</td>
</tr>
<tr>
<td>GSEATests-class</td>
<td>23</td>
</tr>
<tr>
<td>hushWarning</td>
<td>23</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>24</td>
</tr>
<tr>
<td>impute</td>
<td>24</td>
</tr>
<tr>
<td>influence.bayesglm</td>
<td>25</td>
</tr>
<tr>
<td>invlogit</td>
<td>26</td>
</tr>
<tr>
<td>LMERlike-class</td>
<td>26</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMlike-class</td>
<td>27</td>
</tr>
<tr>
<td>logFC</td>
<td>30</td>
</tr>
<tr>
<td>logmean</td>
<td>31</td>
</tr>
<tr>
<td>LRT</td>
<td>32</td>
</tr>
<tr>
<td>lrTest</td>
<td>33</td>
</tr>
<tr>
<td>lrTest,ZlmFit.character-method</td>
<td>34</td>
</tr>
<tr>
<td>magic_assay_names</td>
<td>35</td>
</tr>
<tr>
<td>maits</td>
<td>36</td>
</tr>
<tr>
<td>MAST-defunct</td>
<td>36</td>
</tr>
<tr>
<td>mast_filter</td>
<td>37</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&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>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>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.summaryZlmFit</td>
<td>47</td>
</tr>
<tr>
<td>read.fluidigm</td>
<td>47</td>
</tr>
<tr>
<td>removeResponse</td>
<td>49</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>show,LMlike-method</td>
<td>51</td>
</tr>
<tr>
<td>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,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>vbeta</td>
<td>60</td>
</tr>
<tr>
<td>vbetaFA</td>
<td>60</td>
</tr>
<tr>
<td>waldTest</td>
<td>61</td>
</tr>
<tr>
<td>waldTest,ZlmFit.matrix-method</td>
<td>61</td>
</tr>
<tr>
<td>xform</td>
<td>61</td>
</tr>
<tr>
<td>zlm</td>
<td>62</td>
</tr>
<tr>
<td>ZlmFit-class</td>
<td>64</td>
</tr>
</tbody>
</table>

## Index

<table>
<thead>
<tr>
<th>Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>67</td>
</tr>
</tbody>
</table>
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

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

## 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 ('continuous' or 'discrete') 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
e.example(gseaAfterBoot)

---

colData<-,SingleCellAssay,DataFrame-method
Replace colData

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

## 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, $Y - E(U) \cdot E(V)$
computeEtFromCt

**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.
CovFromBoots

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 'location', 'scale' and degrees of freedom (df), second dimension giving the component ('Continuous' or 'Discrete') 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.Condition\text{Unstim}')
## 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)
```
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.

Description

Puts log transformed values onto natural scale and takes mean of vector. Calculates 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**
filt<e>erLowExpressedGenes(assay, threshold = 0.1)

**Arguments**
- **assay**: a `SingleCellAssay` object
- **threshold**: a numeric between 0, and 1, specifying the threshold frequency below which genes will be filtered out

**Value**
`SingleCellAssay`

**Examples**
```r
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, ...)

```r
## 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)
numexp(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

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
class desired subclass of object. Default SingleCellAssay.
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
Value

SingleCellAssay, or derived, object

Examples

data(vbeta)
colnames(vbeta)
vbeta <- computeEtFromCt(vbeta)
vbeta.fa <- FromFlatDF(vbeta, idvars=c("Subject.ID", "Chip.Number", "Well"),
primerid="Gene", measurement="Et", ncells="Number.of.Cells",
geneid="Gene",cellvars=c('Number.of.Cells', 'Population'),
phenovars=c('Stim.Condition', 'Time'), id='vbeta all', class='FluidigmAssay')
show(vbeta.fa)
nrow(vbeta.fa)
ncol(vbeta.fa)
head(mcols(vbeta.fa)$primerid)
table(colData(vbeta.fa)$Subject.ID)
vbeta.sub <- subset(vbeta.fa, Subject.ID=="Sub01")
show(vbeta.sub)

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

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

Arguments

exprsArray      matrix, or a list of matrices, or an array. Columns are cells, rows are genes.
cData           cellData an object that can be coerced to a DataFrame, ie, data.frame, AnnotatedDataFrame. Must have as many rows as ncol(exprsArray)
fData           featureData an object that can be coerced to a DataFrame, ie, data.frame, AnnotatedDataFrame. Must have as many rows as nrow(exprsArray).
class           desired subclass of object. Default SingleCellAssay.
getConcordance

### 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

```r
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(): getrc 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)
### GLMlike-class

#### 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

#### 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

```r
gseaAfterBoot(
  zFit,
  boots,
  sets,
  hypothesis,
  control = 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 expression. 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`. 

---

### gseaAfterBoot

**Gene set analysis for hurdle model**

---

**gseaAfterBoot**

**Gene set analysis for hurdle model**

---
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**

Describe a linear model hypothesis to be tested

**Description**

A 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
example(stat_ell)

\[\text{influence.bayesglm} \quad \text{Influence bayesglm object}\]

Description

The influence function

Usage

\[
\text{## S3 method for class 'bayesglm'}
\text{influence(model, do.coef = TRUE, ...)}
\]

Arguments

model bayesglm

do.coef see influence.glm

... ignored

Value

see influence.glm
invlogit

Description

Inverse of logistic transformation

Usage

invlogit(x)

Arguments

x numeric

Value

numeric

Examples

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

## 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)
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?
...
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

## 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?
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
- `waldTest(object = LMlike, hypothesis = CoefficientHypothesis)`: Wald test dropping single term specified by `CoefficientHypothesis` hypothesis.
- `waldTest(object = LMlike, hypothesis = matrix)`: Wald test of contrast specified by contrast matrix hypothesis.
- `lrTest(object = LMlike, hypothesis = character)`: Likelihood ratio test dropping entire term specified by character hypothesis naming a term in the symbolic formula.
- `lrTest(object = LMlike, hypothesis = CoefficientHypothesis)`: Likelihood ratio test dropping single term specified by `CoefficientHypothesis` hypothesis.
- `lrTest(object = LMlike, hypothesis = Hypothesis)`: Likelihood ratio test dropping single term specified by `Hypothesis` hypothesis.
- `lrTest(object = LMlike, hypothesis = matrix)`: Likelihood ratio test dropping single term specified by contrast matrix hypothesis.
- `logLik(GLMlike)`: return the log-likelihood of a fitted model.

**Slots**

- `design`: a data.frame from which variables are taken for the right hand side of the regression.
- `fitC`: The continuous fit.
- `fitD`: The discrete fit.
- `response`: The left hand side of the regression.
- `fitted`: A logical with components “C” and “D”, TRUE if the respective component has converged.
- `formula`: A formula for the regression.
- `fitArgsC`:
- `fitArgsD`: Both lists giving arguments that will be passed to the fitter (such as convergence criteria or case weights).

**See Also**

- `coef`
- `lrTest`
- `waldTest`
- `vcov`
- `logLik`
logFC  

*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 contrast0 and the state(s) given by contrast1.

**Usage**

```r
logFC(zlmfit, contrast0, contrast1)
getLogFC(zlmfit, contrast0, contrast1)
```

**Arguments**

- `zlmfit`: ZlmFit output
- `contrast0`: vector of coefficients giving baseline contrast, or a `Hypothesis`. If missing, then the '(Intercept)' is used as baseline.
- `contrast1`: matrix of coefficients giving comparison contrasts, or a `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 contrast0 to 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 logFC. An approximation of the variance of logFC (applying the delta method to formula defined above) is provided in 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**

- `getLogFC()`: Return results as a perhaps friendlier data.table
logmean

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/ 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
coefnames <- colnames(coef(zz, 'D'))
contrast0 <- setNames(rep(0, length(coefnames)), coefnames)
contrast0[c('(Intercept)', 'Stim.ConditionUnstim')] <- 1
contrast1 <- diag(length(coefnames))
rownames(contrast1)<-colnames(contrast1)<-coefnames
  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[,2]==-lfcUnstim$logFC[,1])))
  ##As a data.table:
getLogFC(zz)

logmean

Description

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

\texttt{logmean(x)}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} \text{numeric}
\end{itemize}

Value

\begin{itemize}
  \item \text{numeric}
\end{itemize}

Examples

\begin{verbatim}
x <- 1:10
expavg(logmean(x))
\end{verbatim}

\hrulefill

\textbf{LRT} \hspace{1cm} \textit{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

\begin{verbatim}
LRT(sca, comparison, ...)
\end{verbatim}

\begin{verbatim}
## S4 method for signature 'SingleCellAssay,character'
LRT(sca, comparison, referent = NULL, groups = NULL, returnall = FALSE)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{sca} \hspace{1cm} \text{A SingleCellAssay class object}
  \item \texttt{comparison} \hspace{1cm} \text{A character specifying the factor for comparison}
  \item \texttt{...} \hspace{1cm} \text{ignored}
  \item \texttt{referent} \hspace{1cm} \text{A character specifying the reference level of comparison.}
  \item \texttt{groups} \hspace{1cm} \text{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.}
  \item \texttt{returnall} \hspace{1cm} \text{A logical specifying if additional rows should be returned with information about the different components of the test.}
\end{itemize}

Details

Combined Likelihood ratio test (binomial and normal) for SingleCellAssay and derived objects. This function is deprecated, please use \texttt{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
eexample('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

## 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.

pool: 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).

e`exprs_values`: character or integer passed to `assay` specifying which assay to use for testing

Value

3D array
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             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**

```r
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)
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, ...)
```
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, ...)
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

```r
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

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

Description

Sample cells with replacement to find bootstrapped distribution of coefficients

Usage

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

data(vbetaFA)
zlmVbeta <- zlm(~ Stim.Condition, subset(vbetaFA, ncells==1)[1:5,])
#Only run 3 boot straps, 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

## 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

## 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".
plotSCAConcordance

**Usage**

```r
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 `mcols` column `ncells` is set then the averages will be adjusted accordingly. But it could be any grouping.

**Usage**

```r
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)
<table>
<thead>
<tr>
<th>predicted_sig</th>
<th>Predicted signatures</th>
</tr>
</thead>
</table>

**Description**

Predicted signatures

**Format**

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

<table>
<thead>
<tr>
<th>primerAverage</th>
<th>Average expression values for duplicated/redundant genes</th>
</tr>
</thead>
</table>

**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**

- `fd`: SingleCellAssay or subclass
- `geneGroups`: character naming a column in the featureData that keys the duplicates
- `fun.natural`: transformation to be used to collapse the duplicate expression values
- `fun.cycle`: transformation to be used after collapsing

**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

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.
**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
Description
rstandard bayesglm object S3 method

Usage
## 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
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.

se.coef

Return coefficient standard errors

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
df <- example('ZlmFit-class')

show,LMlike-method

show

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

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

Inherit aesthetics

If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

Fill

A color or aesthetic mapping to fill color. Defaults to NA for empty ellipses.

Level

The confidence level at which to draw an ellipse (default is level=0.95).

Lty

The linetype to use. Can map to a variable. Defaults to 2 (dashed line)

Invert

Vector of length 1 that should either be "x", "y", or TRUE. Specifies whether to plot the estimates from the discrete component on the inverse logit scale. invert specifies which axis to invert.

Alpha

Transparency

... other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

Value

ggplot layer

Examples

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, model.matrix=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')

subset,SingleCellAssay-method

Subset a SingleCellAssay by cells (columns)

Description

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

Usage

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

Arguments

- `x` SingleCellAssay
- `...` expression

Value

SingleCellAssay

Examples

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

Description

Return programmatically useful summary of a fit

Usage

```r
summarize(object, ...)
```

Arguments

- `object` LMlike or subclass
- `...` other arguments

Value

list of parameters characterizing fit
### Summary

**GSEATests**

_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 (ie, 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
example(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
Value

data.table

See Also

print.summaryZlmFit

Examples

data(vbetaFA)
z <- zlm(~Stim.Condition, vbetaFA[1:5,])
zs <- summary(z)
names(zs)
print(zs)
##Select `datatable` copmonent to get normal print method
zs$datatable
## Can use parallel processing for LRT now
summary(z, doLRT = TRUE, parallel = TRUE)
Functions

- `print(summaryThresholdSCRNA)`: prints five-number distillation of the statistics and invisibly returns the table used to generate the summary

### 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

```r
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)

---

vbmeta 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.

---

vbmetaFA 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
table('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

---

`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

```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`
Usage

```
zlm(
  formula, 
  sca, 
  method = "bayesglm", 
  silent = TRUE, 
  ebayes = TRUE, 
  ebayesControl = NULL, 
  force = FALSE, 
  hook = NULL, 
  parallel = TRUE, 
  LMlike, 
  onlyCoef = FALSE, 
  exprs_values = assay_idx(sca)$aidx, 
  ...
)
```

Arguments

- `formula`: a formula with the measurement variable on the LHS and predictors present in `colData` on the RHS.
- `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.
- `...`: arguments passed to the S4 model object upon construction. For example, `fitArgsC` and `fitArgsD`, or `coefPrior`.

Value

A object of class `ZlmFit` with methods to extract coefficients, etc. OR, if data is a `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

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

## Can also provide just a \code{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'
v cov(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.
Slots

- `coefC` matrix of continuous coefficients
- `coefD` 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 pseudo-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')
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'))

# 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
dof, GLMlike-method (dof), 11
dof, LMERlike-method (dof), 11
Drop, 12
ebayes, 12, 34, 63
expavg, 13, 31
expres (MAST-defunct), 36
fit, BayesGLMlike, missing-method (fit), 14
fit, GLMlike, 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
getric (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
INDEX

influence.glm, 25
invlogit, 26
LMERlike-class, 26
LMlike-class, 27
logFC, 30
logLik, GLMlike-method
LMERlike-class, 26
logLik, LMlike-class, 27
logFC, 30
logLik, LMERlike-method
logmean, 31
LRT, 32
LRT, SingleCellAssay, character-method
lrTest, 32, 33
lrTest, LMlike, character-method
lrTest, LMlike, CoefficientHypothesis-method
lrTest, LMlike, Hypothesis-method
lrTest, LMlike, matrix-method
lrTest, ZlmFit, character-method
lrTest, ZlmFit, CoefficientHypothesis-method
lrTest, ZlmFit, Hypothesis-method
lrTest, ZlmFit, matrix-method
magic_assay_names, 35
maits, 36
MAST (MAST-package), 4
MAST-defunct, 36
MAST-package, 4
mcols, 36
meld_list_left, 38
melt.SingleCellAssay, 39
model.matrix, 39
model.matrix, LMlike-method
(model.matrix), 39
model.matrix <-, 40
myBiplot, 40
new_with_repaired_slots, 41
numexp(freq), 15
partialScore(collectResiduals), 7
pbootVcov1, 42
plot.thresholdSCRNACountMatrix, 43
plotlrt, 43
plotSCAConcordance, 19, 44
predict.ZlmFit, 45
predicted_sig, 46
primerAverage, 46
print.summaryThresholdSCRNA
   (summary.thresholdSCRNACountMatrix), 58
print.summaryZlmFit, 47
rbind, 36
read.fluidigm, 47
read.table, 48
removeResponse, 49
rstandard, 50
rstandard.bayesglm, 50
SceToSingleCellAssay, 50
se.coef, 51
se.coef, ZlmFit-method (ZlmFit-class), 64
show, LMlike-method, 51
show, ZlmFit-method
   (show, LMlike-method), 51
SingleCellAssay (FromFlatDF), 16
split, SingleCellAssay, character-method,
   52
split, SingleCellAssay, factor-method
   (split, SingleCellAssay, character-method), 52
split, SingleCellAssay, list-method
   (split, SingleCellAssay, character-method),
   52
stat_ell, 53
subset, SingleCellAssay-method, 54
summarize, 55
summary, GSEA_tests-method, 56
summary, LMlike-method (LMlike-class), 27
summary, ZlmFit-method, 37, 57
summary-ZlmFit,
   (summary.ZlmFit-method), 57
summary.thresholdSCRNAcountMatrix, 58
thresholdSCRNAcountMatrix, 59
update, LMERlike-method
   (LMERlike-class), 26
update, LMlike-method (LMlike-class), 27
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