Package ‘MAST’

November 11, 2016

Type Package
Title Model-based Analysis of Single Cell Transcriptomics
Version 1.0.0
Date 2016-10-10
Author Andrew McDavid <Andrew_McDavid@urmc.rochester.edu>, Greg Finak <gfinak@fredhutch.org>, Masanao Yajima <myajima@fredhutch.org>
Maintainer Andrew McDavid <Andrew_McDavid@urmc.rochester.edu>
VignetteBuilder knitr
Imports Biobase, BiocGenerics, S4Vectors, data.table, ggplot2, plyr, stringr, abind, methods, parallel, reshape2, stats, stats4, graphics, utils
Depends SummarizedExperiment, R(>= 3.3)
Suggests knitr, rmarkdown, testthat, lme4(>= 1.0), roxygen2(> 4.0.0), numDeriv, car, gdata, lattice, GGally, GSEABase, NMF, TxDb.Hsapiens.UCSC.hg19.knownGene, rsvd, limma, RColorBrewer
Description Methods and models for handling zero-inflated single cell assay data.
License GPL(>= 2)
Collate 'AllGenerics.R' 'AllClasses.R' 'Fluidigm-methods.R'
 'GSEA-by-boot.R' 'Hypothesis.R' 'LmWrapper.R'
 'MultidimensionalScaling.R' 'RNASeqAssay-methods.R' 'Readers.R'
 'SingleCellAssay-methods.R' 'UtilityFunctions.R'
 'ZlmFit-bootstrap.R' 'ZlmFit-logFC.R' 'ZlmFit.R' 'bayesglm.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 5.0.1
LazyData true
biocViews GeneExpression, DifferentialExpression, GeneSetEnrichment, RNASeq, Transcriptomics, SingleCell
NeedsCompilation no
R topics documented:

MAST-package ........................................... 3
applyFlat .................................................. 4
BayesGLMlike-class ..................................... 4
bootVcov1 ................................................ 5
calcZ ...................................................... 5
cData ...................................................... 6
colData<-.SingleCellAssay,DataFrame-method ............ 7
collectResiduals .......................................... 8
computeEtFromCt ........................................ 10
defaultPrior ............................................ 10
dof ........................................................ 12
Drop ........................................................ 12
ebayes ...................................................... 13
expavg ..................................................... 13
fData ...................................................... 14
featureData .............................................. 14
filter ....................................................... 14
filterLowExpressedGenes .................................. 16
fit ........................................................ 16
freq ........................................................ 17
FromFlatDF ............................................... 17
FromMatrix ............................................... 18
getConcordance ......................................... 19
getwellKey ............................................... 20
GLMlike-class ........................................... 21
gseaAfterBoot .......................................... 22
GSEATests-class ........................................ 23
hushWarning ............................................ 23
Hypothesis .............................................. 24
impute ..................................................... 25
influence.bayesglm ..................................... 25
invlogit ................................................... 26
LMERlike-class ........................................ 26
LMlike-class ............................................ 27
logFC ...................................................... 29
logmean ................................................... 30
LRT ........................................................ 31
lrTest ..................................................... 32
lrTest,ZlmFit,character-method ......................... 32
maits ...................................................... 33
melt.SingleCellAssay ................................... 33
model.matrix ........................................... 34
model.matrix<- ......................................... 34
myBiplot ................................................. 35
numexp ................................................... 35
pbootVcov1 ............................................. 36
plot.thresholdSCRNACountMatrix ......................... 36
plotlrt .................................................. 37
MAST-package

Description

Methods for analysing single cell assay data using hurdle models.

Details

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

References

**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 = "+")
```

**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?
bootVcov1

**Bootstrap a zlmfit**

**Description**

Sample cells with replacement to find bootstrapped distribution of coefficients

**Usage**

```r
bootVcov1(zlmfit, R = 99)
```

**Arguments**

- `zlmfit`: class `ZlmFit`
- `R`: number of bootstrap replicates

**Value**

array of bootstrapped coefficients

**Examples**

```r
data(vbetaFA)
zlmVbeta <- zlm.SingleCellAssay(~ 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)
```

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**

```r
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

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

Description

These functions are now all deprecated and will be removed in a future release.

Usage

```r
cData(sc)
cData(sc) <- value

## S4 method for signature 'SingleCellAssay'
cData(sc)

## S4 replacement method for signature 'SingleCellAssay'
cData(sc) <- value

## S4 method for signature 'SingleCellAssay,SingleCellAssay'
combine(x, y, ...)

## S4 method for signature 'SingleCellAssay,ANY'
combine(x, y, ...)
```
Arguments

sc  An object with cellData
value  replacement value
x  SingleCellAssay
y  SingleCellAssay
...  SingleCellAssay

Details

cData(sc): Return the cellData data.frame.
cData(sc)<-value: Replace the cellData with value, which can be either an AnnotatedDataFrame or data.frame. The replacement is checked that it has mandatory fields defined by its class.
combine(x, y, ...): Concatenate two experiments along rows/columns

Value

DataFrame or modifies the SingleCellAssay object in place

Replacement Functions

You should transition to use the following replacements:
cData  colData
fData  mcols
exprs  assay
combine  cbind2 or rbind2

See Also

exprs

Examples

data(vbetaFA)
stopifnot(all.equal(hushWarning(cData(vbetaFA), 'deprecated'), colData(vbetaFA)))
stopifnot(all.equal(hushWarning(fData(vbetaFA), 'deprecated'), mcols(vbetaFA)))
stopifnot(all.equal(hushWarning(exprs(vbetaFA), 'deprecated'), t(assay(vbetaFA))))

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.
collectResiduals

Usage

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

Arguments

- `x` SingleCellAssay
- `value` DataFrame

Value

modified SingleCellAssay

---

collectResiduals  Residual hooks and collection methods

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, i.e., difference between expected probability of expression and observed.

- `continuous_residuals_hook`: Hook to get the continuous residuals, i.e., residuals for conditionally positive observations. If an observation is zero, its residual is defined to be zero as well.

- `combined_residuals_hook`: Hook to get the combined residuals, i.e., $Y - E(U)E(V)$.

- `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 $\hat{p}$, 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.SingleCellAssay`

Examples

data(vbetaFA)
svbeta <- subset(vbetaFA, ncells==1)
svbeta <- svbeta[height(svbeta)>.4,]
window <- function(x1) lapply(assays(x1), function(x2) x2[1:3, 1:6])
# total residuals of the response
z1 <- zlm.SingleCellAssay(~ Stim.Condition, svbeta, hook=discrete_residuals_hook)
window(collectResiduals(z1, svbeta))
z2 <- zlm.SingleCellAssay(~ Stim.Condition, svbeta, hook=continuous_residuals_hook)
window(collectResiduals(z2, svbeta))
z3 <- zlm.SingleCellAssay(~ Stim.Condition, svbeta, hook=combined_residuals_hook)
window(collectResiduals(z3, svbeta))
# total deviance residuals
z4 <- zlm.SingleCellAssay(~ Stim.Condition, svbeta, hook=deviance_residuals_hook)
window(collectResiduals(z4, svbeta))
# partial residuals
colData(svbeta)$ngeneson <- colMeans(assay(svbeta)>0)
z5 <- zlm.SingleCellAssay(~ 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.

**Usage**

```r
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**

```r
data(vbeta)
vbeta <- computeEtFromCt(vbeta)
```

---

**condmean**  
*Report the mean et value for each gene*

**Description**

NAs are always removed.

**Usage**

```r
condmean(sc)
```

**Arguments**

- `sc`: SingleCellAssay

**Value**

vector of means
Examples

```r
data(vbetaFA)
condmean(vbetaFA)
```

---

**condSd**

Report standard deviation of et, for positive et for each gene

### Description

NAs are always removed

### Usage

```r
condSd(sc)
```

### Arguments

- `sc`  
  SingleCellAssay

### Value

vector of standard deviations

---

**defaultPrior**

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

### Description

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

### Usage

```r
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 (‘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

```r
dp <- defaultPrior('Stim.ConditionUnstim')
## Not run:
data(vbetaFA)

## End(Not run)
```
**dof**

*Degrees of freedom of Zero inflated model*

**Description**

Degrees of freedom of Zero inflated model

**Usage**

```r
dof(object)
```

**Arguments**

- `object` LMlike or subclass

**Value**

A vector giving the model degrees of freedom for continuous and discrete parts.

---

**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 `formula`.

**Usage**

```r
ebayes(sca, ebayesControl, Formula, truncate = Inf)
```

**Arguments**

- `sca` SingleCellAssay
- `ebayesControl` list with (optional) components 'method', 'model'. See details.
- `Formula` a formula (using variables in `colData(sca)` used when `model='Var'`.
- `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 mean(2^x - 1)

**Usage**

```r
expavg(x)
```

**Arguments**

- `x` numeric

**Value**

numeric

**Examples**

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

Description
Accessor for featureData data.frame

Arguments
object An object with featureData

Details
Returns the featureData data.frame.

Value
data.frame

featureData

Description
Returns the featureData.

Arguments
object An object with featureData

Value
AnnotatedDataFrame

filter
Filter a SingleCellAssay

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

Usage
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 <- filter(vbetaFA, groups='ncells')
## Returned as boolean matrix
was.filtered <- 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')
**filterLowExpressedGenes**

*Filter low-expressing genes*

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

**Usage**
```r
filterLowExpressedGenes(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**
```r
fit(object, response, ...)  
```

### S4 method for signature 'LMERlike,missing'
```r
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

**Value**
`LMlike` or subclass
freq

Report the proportion of expression for each gene

Description

NAs can be optionally removed

Usage

freq(sc, na.rm = TRUE)

Arguments

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

Value

vector of proportions

Examples

data(vbetaFA)
freq(vbetaFA)

FromFlatDF

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 Summa-
rizedExperiment 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", ...)
FluidigmAssay(...)
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")
getConcordance

Arguments

- `exprsArray` matrix or 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).

Value

an object of class `class`

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

Description

Get the concordance between two

Usage

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

gtwss(concord, nexp)

gtss(concord)

gtrc(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

Details

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.

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)

---

getwellKey  

Accessor for wellKey

Description

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

Usage

getwellKey(sc)
Arguments

sc             A object with a wellKey

Value

integer giving the unique id generated

Examples

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

GLMlike-class   Wrapper for regular glm/lm

Description

Wrapper for regular glm/lm

Usage

## 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: return the variance/covariance of component which

Slots

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

Gene set analysis for hurdle model

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 = list(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` list of control parameters. See details.

Value

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

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.SingleCellAssay(~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'))
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)
Hypothesis

**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')

---

**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.SingleCellAssay waldTest lrTest

**Examples**

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**

```r
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**

```r
## See stat_ell
example(stat_ell)
```

---

**influence.bayesglm**

*Influence bayesglm object*

**Description**

The influence function

**Usage**

```r
## 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**
`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, ...)

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

Slots

pseudoMM  part of this horrendous hack.

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, ...)

## 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
- **...**: passed to model.matrix
- **hypothesis**: one of a CoefficientHypothesis, Hypothesis or contrast matrix.

### Value

see section "Methods (by generic)"

### Methods (by generic)

- **summary**: Print a summary of the coefficients in each component.
- **update**: update the formula or design from which the model.matrix is constructed
- **waldTest**: Wald test dropping single term specified by CoefficientHypothesis hypothesis
- **waldTest**: Wald test of contrast specified by contrast matrix hypothesis
- **lrTest**: Likelihood ratio test dropping entire term specified by character hypothesis naming a term in the symbolic formula.
- **lrTest**: Likelihood ratio test dropping single term specified by CoefficientHypothesis hypothesis
- **lrTest**: Likelihood ratio test dropping single term specified by Hypothesis hypothesis
- **lrTest**: Likelihood ratio test dropping single term specified by contrast matrix hypothesis
- **logLik**: 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**: Both lists giving arguments that will be passed to the fitter (such as convergence criteria or case weights)


See Also

decomp
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

logFC(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 ex-
pected 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 \( \text{varLogFC} \).

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

• getLogFC: Return results as a perhaps friendlier data.table
See Also

Hypothesis

Examples

data(vbetaFA)
## 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
coefficientNames <- colnames(coef(zz, 'D'))
contrast0 <- setNames(rep(0, length(coefficientNames)), coefficientNames)
contrast0[c('Intercept', 'Stim.ConditionUnstim')] <- 1
contrast1 <- diag(length(coefficientNames))
rownames(contrast1) <- colnames(contrast1) <- coefficientNames
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.

Value

data.frame

See Also

zlm.SingleCellAssay, ZlmFit

Examples

data(vbetaFA)
LRT(vbetaFA, 'Stim.Condition', 'Unstim')
**lrTest**  
*Run a likelihood-ratio test*

**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**

```r
lrTest(object, hypothesis)
```

**Arguments**

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

**Value**

array giving test statistics

**See Also**

`fit`  
`waldTest`  
`Hypothesis`  
`CoefficientHypothesis`

**Examples**

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

---

**lrTest,ZlmFit,character-method**

*Likelihood ratio 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,character'
lrTest(object, hypothesis)
```
### maits

**Arguments**
- object: ZlmFit
- hypothesis: See Details

**Value**
- 3D array

---

**Description**
MAITs data set, RNASeq

**Format**
- a list containing an expression matrix (expressionmat), cell cdat and feature fdat.

**See Also**
- FromMatrix

---

**melt.SingleCellAssay**  
*Melt a rectangular array*

**Description**
Return a molten (flat) representation of a rectangular array

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

**Arguments**
- data: A rectangular array, with attributes attached to its rows and columns
- ...: ignored
- na.rm: ignored
- value.name: name of 'values' column containing the measurement

**Value**
A data.frame typically, with the cartesian product of the row and column attributes and the values from the rectangular array

**Examples**
```r
data(vbetaFA)
as(vbetaFA[1:10,], 'data.table')
```
model.matrix

Model matrix accessor

Description
Model matrix accessor

Usage
model.matrix(object, ...)

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

Arguments

object LMlike or subclass

... ignored

Value
model.matrix if present

Methods (by class)

- LMlike: return the model.matrix

model.matrix<-

Replace 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**

```r
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

---

**numexp**

*Report number of expressing cells per gene*

**Description**

NAs are removed

**Usage**

```r
numexp(sc)
```

**Arguments**

- `sc` SingleCellAssay

**Value**

numeric vector
**pbootVcov1**

*Bootstrap a zlmfit*

**Description**

Sample cells with replacement to find bootstrapped distribution of coefficients.

**Usage**

```r
tpbootVcov1(cl, zlmfit, R = 99)
```

**Arguments**

- `cl`: a cluster object created by `makeCluster`
- `zlmfit`: class `ZlmFit`
- `R`: number of bootstrap replicates

**Value**

array of bootstrapped coefficients

---

**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`
**plotlrt**

**Value**

plots displays plots

---

**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**

```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 p-values 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, ...)
```
predict.ZlmFit

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`

Value

- printed plot

See Also

- `getConcordance`

Examples

```r
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

```r
## 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 and standard errors.

Examples

```r
##See stat_ell
text(stat_ell)
```
primerAverage

**Description**

This is broken at the moment (transposition errors).

**Usage**

```r
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**

collapsed version of `fd`.

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`
Description

Reads a fluidigm raw data file (or set of files)

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.
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".
**removeResponse**

- **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.

**Details**

This function reads a raw Fluidigm data file or set of files and constructs a SingleCellAssay (or FluidigmAssay) object.

**Value**

A list of SingleCellAssay holding the data.

**Author(s)**

Greg Finak

---

**removeResponse**

Remove the left hand side (response) from a formula

**Description**

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

**Usage**

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

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
example("ZlmFit-class")
### show,LMlike-method

#### Description
Display info

#### Usage

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

## S4 method for signature 'ZlmFit'
show(object)
```

#### Arguments

- `object`: an object of some type

#### Details

Prints information on a LMlike object

#### Value

side effect of printing to console

### Methods (by class)

- **ZlmFit**: print info on ZlmFit

### split,SingleCellAssay,character-method

#### 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

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, ...)

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.

invert.aes 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).
**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**

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

**Arguments**

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

**Value**

SingleCellAssay

---

**Examples**

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

zlmCond <- glm(~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(muD), 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=muD, 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")
```
Examples

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

summarize

Return programmatically useful summary of a fit

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-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 (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)
```

---

### 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, ...)
```

### 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.
- ... 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' copmoment to get normal print method
zs$datatable

summary.thresholdSCRNACountMatrix

Summarize 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

Methods (by generic)

• print: prints five-number distillation of the statistics and invisibly returns the table used to generate the summary
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.

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 bandwidth adjustment, passed to density

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

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.
waldTest

---

**Description**

Vbeta Data Set, FluidigmAssay

**Format**

a FluidigmAssay of the vbeta data set.

**See Also**

vbeta, FromFlatDF

---

waldTest

---

**Description**

Run a Wald test on discrete and continuous components. The 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

---

**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
Convenience function for running a zero-inflated regression

Description
Fits a hurdle model on zero-inflated continuous data in which the zero process is modeled as a logistic regression and (conditional on the the response being >0), the continuous process is Gaussian, i.e., a linear regression.

Usage
zlm(formula, data, method = "bayesglm", silent = TRUE, ...)

Arguments
- **formula**: model formula
- **data**: a data.frame, list, environment or SingleCellAssay in which formula is evaluated
- **method**: one of 'glm', 'glmer' or 'bayesglm'. See MAST:::methodDict for other possibilities.
- **silent**: if TRUE suppress common errors from fitting continuous part
- **...**: passed to fit, and eventually to the linear model fitting function

Value
list with "disc"rete part and "cont"inuous part

See Also
GLMlike, LMERlike, BayesGLMlike

Examples
```r
data<- data.frame(x=rnorm(500), z=rbinom(500, 1, .3))
logit.y <- with(data, x*2 + z*2); 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)
summary.glm(fit$cont)
```
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.SingleCellAssay(formula, sca, method = "bayesglm", silent = TRUE, 
                    ebayes = TRUE, ebayesControl = NULL, force = FALSE, hook = NULL, 
                    parallel = TRUE, LMlike, onlyCoef = FALSE, ...)

Arguments
  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).  

... 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.

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
ZlmFit-class

An S4 class to hold the output of a call to zlm

Description

This holds output from a call to zlm.SingleCellAssay. Many methods are defined to operate on it. See below.

Usage

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

Examples

```r
data(vbetaFA)
zlmVbeta <- zlm.SingleCellAssay(~ 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'))
```
Arguments

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

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
device 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

See Also

zlm.SingleCellAssay summary,ZlmFit-method
Examples

data(vbetaFA)
zlmVbeta <- zlm.SingleCellAssay(~ 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, that the columns must be enclosed in backquotes to protect the \
lrTest(zlmVbeta, Hypothesis("\'PopulationCD154+VbetaUnresponsive' - 'PopulationCD154-VbetaUnresponsive'"))
waldTest(zlmVbeta, Hypothesis("\'PopulationCD154+VbetaUnresponsive' - 'PopulationCD154-VbetaUnresponsive'"))
Index

applyFlat, 4
assay, 7
BayesGLMlike-class, 4
bootVcov1, 5
burdenOfFiltering (filter), 14
calcZ, 5
cbind2, 7
cData, 6
cData,SingleCellAssay-method (cData), 6
cData<- (cData), 6
cData<-,SingleCellAssay-method (cData), 6
coef,LMERlike-method (LMERlike-class), 26
coef,ZlmFit-method (ZlmFit-class), 54
CoefficientHypothesis, 55
CoefficientHypothesis (Hypothesis), 24
coldData, 7
coldData<-,SingleCellAssay,DataFrame-method, 7
collectResiduals, 8
combine,SingleCellAssay,ANY-method (cData), 6
combine,SingleCellAssay,SingleCellAssay-method (cData), 6
combined_residuals_hook (collectResiduals), 8
computeEtFromCt, 10
condmean, 10
condSd, 11
continuous_residuals_hook (collectResiduals), 8
defaultPrior, 11
deviance_residuals_hook (collectResiduals), 8
discrete_residuals_hook (collectResiduals), 8
dof, 12
dof,LMlike-method (dof), 12
Drop, 12
ebayes, 13, 53
expavg, 13, 30
fData, 14
fData,SingleCellAssay-method (fData), 14
featureData, 14
featureData,SingleCellAssay-method (featureData), 14
filter, 14
filterLowExpressedGenes, 16
fit, 16
fit,LMlike,missing-method (fit), 16
fit,LMERlike,missing-method (fit), 16
fitted_phat (collectResiduals), 8
FluidigmAssay, 17
FluidigmAssay (FromFlatDF), 17
freq, 17
FromFlatDF, 17, 50
FromMatrix, 18, 33
getConcordance, 19
getLogFC (logFC), 29
getss (getConcordance), 19
getwellKey, 20
getwellKey,SingleCellAssay-method (getwellKey), 20
getwss (getConcordance), 19
GLMlike-class, 21
gseaAfterBoot, 22
GSEATests-class, 23
hushWarning, 23
Hypothesis, 24, 29, 55
impute, 25
influence.bayesglm, 25
influence.glm, 25
invlogit, 26
LMERlike-class, 26
LMlike-class, 27
logFC, 29
logLik,LMlike-method (LMlike-class), 27
logLik,LMERlike-method (LMERlike-class), 26
<table>
<thead>
<tr>
<th>Logmean</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRT</td>
<td>31</td>
</tr>
<tr>
<td>LRT, SingleCellAssay, character-method (LRT)</td>
<td>31</td>
</tr>
<tr>
<td>lrTest, 31, 32</td>
<td></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, 32</td>
<td></td>
</tr>
<tr>
<td>lrTest, ZlmFit, CoefficientHypothesis-method (ZlmFit-class)</td>
<td>54</td>
</tr>
<tr>
<td>lrTest, ZlmFit, Hypothesis-method (ZlmFit-class)</td>
<td>54</td>
</tr>
<tr>
<td>lrTest, ZlmFit, matrix-method (ZlmFit-class)</td>
<td>54</td>
</tr>
<tr>
<td>maits</td>
<td>33</td>
</tr>
<tr>
<td>MAST (MAST-package)</td>
<td>3</td>
</tr>
<tr>
<td>MAST-package</td>
<td>3</td>
</tr>
<tr>
<td>mcols</td>
<td>7</td>
</tr>
<tr>
<td>melt.SingleCellAssay</td>
<td>33</td>
</tr>
<tr>
<td>model.matrix</td>
<td>34</td>
</tr>
<tr>
<td>model.matrix, LMlike-method (model.matrix)</td>
<td>34</td>
</tr>
<tr>
<td>model.matrix&lt;-</td>
<td>34</td>
</tr>
<tr>
<td>myBiplot</td>
<td>35</td>
</tr>
<tr>
<td>numexp</td>
<td>35</td>
</tr>
<tr>
<td>partialScore (collectResiduals)</td>
<td>8</td>
</tr>
<tr>
<td>pbootVcov1</td>
<td>36</td>
</tr>
<tr>
<td>plot.thresholdSCRNAcountMatrix</td>
<td>36</td>
</tr>
<tr>
<td>plotlrt</td>
<td>37</td>
</tr>
<tr>
<td>plotSCAConcordance</td>
<td>37</td>
</tr>
<tr>
<td>predict.ZlmFit</td>
<td>38</td>
</tr>
<tr>
<td>primerAverage</td>
<td>39</td>
</tr>
<tr>
<td>print.summaryThresholdSCRNA (summary.thresholdSCRNAcountMatrix)</td>
<td>48</td>
</tr>
<tr>
<td>print.summaryZlmFit</td>
<td>39</td>
</tr>
<tr>
<td>rbind2</td>
<td>7</td>
</tr>
<tr>
<td>read.fluidigm</td>
<td>40</td>
</tr>
<tr>
<td>read.table</td>
<td>40</td>
</tr>
<tr>
<td>removeResponse</td>
<td>41</td>
</tr>
<tr>
<td>rstandard</td>
<td>42</td>
</tr>
<tr>
<td>rstandard.bayesglm</td>
<td>42</td>
</tr>
<tr>
<td>se.coef</td>
<td>42</td>
</tr>
<tr>
<td>se.coef, ZlmFit-method (ZlmFit-class)</td>
<td>54</td>
</tr>
<tr>
<td>show, LMlike-method</td>
<td>43</td>
</tr>
<tr>
<td>show, ZlmFit-method</td>
<td>43</td>
</tr>
<tr>
<td>singleCellAssay</td>
<td>17</td>
</tr>
<tr>
<td>split, SingleCellAssay, character-method</td>
<td>43</td>
</tr>
<tr>
<td>split, SingleCellAssay, factor-method (split, SingleCellAssay, character-method)</td>
<td>43</td>
</tr>
<tr>
<td>split, SingleCellAssay, list-method (split, SingleCellAssay, character-method)</td>
<td>43</td>
</tr>
<tr>
<td>stat_ell</td>
<td>44</td>
</tr>
<tr>
<td>subset, SingleCellAssay-method</td>
<td>45</td>
</tr>
<tr>
<td>summarize</td>
<td>46</td>
</tr>
<tr>
<td>summary, GSEA Tests</td>
<td>46</td>
</tr>
<tr>
<td>summary, LMlike-method</td>
<td>27</td>
</tr>
<tr>
<td>summary, ZlmFit-method</td>
<td>47</td>
</tr>
<tr>
<td>summary, thresholdSCRNAcountMatrix</td>
<td>48</td>
</tr>
<tr>
<td>update, LMERlike-method (LMERlike-class)</td>
<td>26</td>
</tr>
<tr>
<td>vbeta</td>
<td>49, 50</td>
</tr>
<tr>
<td>vbetaFA</td>
<td>50</td>
</tr>
<tr>
<td>vcov, GLM-like-method</td>
<td>21</td>
</tr>
<tr>
<td>vcov, LMER-like-method</td>
<td>26</td>
</tr>
<tr>
<td>vcov, ZlmFit-method</td>
<td>54</td>
</tr>
<tr>
<td>waldTest</td>
<td>50</td>
</tr>
<tr>
<td>waldTest, LMlike-method (LMlike-class)</td>
<td>27</td>
</tr>
<tr>
<td>waldTest, LMlike-method (LMlike-class)</td>
<td>27</td>
</tr>
<tr>
<td>waldTest, ZlmFit-method (ZlmFit-class)</td>
<td>54</td>
</tr>
<tr>
<td>waldTest, ZlmFit, Hypothesis-method (ZlmFit-class)</td>
<td>54</td>
</tr>
<tr>
<td>waldTest, ZlmFit, matrix-method</td>
<td>51</td>
</tr>
<tr>
<td>xform</td>
<td>51</td>
</tr>
<tr>
<td>zlm</td>
<td>52</td>
</tr>
<tr>
<td>zlm.SingleCellAssay</td>
<td>53</td>
</tr>
<tr>
<td>ZlmFit-class</td>
<td>54</td>
</tr>
</tbody>
</table>