Package ‘QuasiSeq’

August 15, 2022

Version 1.0-11-0

Date 2022-08-11

Title Analyzing RNA Sequencing Count Tables Using Quasi-Likelihood

Depends R (>= 4.0.0)

Imports edgeR, mgcv, pracma, utils, grDevices, graphics

Suggests BB, nleqslv

Enhances stats

Description Identify differentially expressed genes in RNA-seq count data using quasi-Poisson or quasi-negative binomial models with 'QL', 'QLShrink' and 'QLSpline' methods described by Lund, Nettleton, McCarthy, and Smyth (2012) <DOI:10.1515/1544-6115.1826>. Report bias-reduced estimates of log fold changes.

License GPL (>= 2)

NeedsCompilation yes

LazyData yes

Biarch yes

RoxygenNote 6.0.1

Author Steve Lund [aut, cre],
Long Qu [aut, ctb],
Klirk Bhasavanich [aut],
Ian Marschner [aut] (The author of glm2::glm.fit2, which was modified
slightly leading to glm.fit3 in this package.),
R Core Team [aut] (The author of stats::glm.fit, which was modified
slightly leading to glm.fit3 in this package.)

Maintainer Steve Lund <`lundsp@gmail.com`>

Repository CRAN

Date/Publication 2022-08-15 11:40:12 UTC

R topics documented:

coef.glm 2

fbrNBglm 3
glmSolvers 5
mockRNASeqData 6
NBDev 7
negbin 9
nlSolvers 10
PoisDev 11
QL.fit 12
QL.results 16

Index **18**

coef.glm — *Coefficient and Bias From Generalized Linear Model Fit*

Description

`coef.glm` is a `S3` method computing the raw coefficients, leading order bias, and bias corrected coefficients from a generalized linear model (glm) fit.

Usage

```
## S3 method for class 'glm'
coef(object, type = c("raw", "bias", "corrected"), ...)
```

Arguments

`object`	An object with similar structure as returned from `glm`.
`type`	A character scalar, being one of `'raw'`, `'bias'`, and `'corrected'`. `'raw'` returns the usual coefficients; `'bias'` returns the estimated leading order bias; and `'corrected'` returns the coefficients after subtracting the estimated bias.
`...`	Not used.

Value

A numeric vector of requested components. When `type='corrected'`, the `'bias'` attribute will be set to a numeric vector of estimated biases being subtracted from the raw coefficients.

Author(s)

Long Qu <rtistician@gmail.com>

References

McCullagh and Nelder (1989) "Generalized Linear Models", 2nd edition. London: Chapman and Hall.

Cordeiro and McCullah (1991) "Bias Correction in Generalized Linear Models", *Journal of the Royal Statistical Society: Series B*, **53**, 629-643.

Examples

```
x=1:30
y=rpois(30L, x/10)
glmfit=glm(y~x, poisson('log'))
coef(glmfit)
coef(glmfit, type='bias')
coef(glmfit, type='corrected')
```

`fbrNBglm` *Firth-Type Bias-Reduced Negative Binomial Log-Linear Model*

Description

This is the estimation algorithm for generalized linear model with negative binomial responses and log link function. Sore functions are modified properly such that bias in the coefficients are reduced.

Usage

```
fbrNBglm.fit(x, y, weights = rep(1, length(y)), offset = rep(0, length(y)),
  family, odisp, control = fbrNBglm.control())

fbrNBglm.control(standardizeX = TRUE, coefOnly = TRUE,
  solvers = nlSolvers, verbose = FALSE, maxit = 25L, start = NULL,
  infoParms = list(j = 1, k = 1, m = 1), order = 2L,
  tol = sqrt(.Machine$double.eps), qr.tol = tol)
```

Arguments

`x, y, weights, offset`	Defined the same as in `glm.fit`.
`family`	The same as in `glm.fit`, but the default (and the only currently supported) choice is `negbin{'log', odisp}`, where `odisp` is defined below.
`odisp`	A numeric scalar of negative binomial over-dispersion parameter. This is the same as `1/size`, where `size` is the parameter as in `dnbinom`.
`control`	A list returned from `fbrNBglm.control`.
`standardizeX`	A logical scalar. If `TRUE`, columns of design matrix will be standardized with norm 1 during the fitting process, except for columns with identical values.
`coefOnly`	A logical scalar. If `TRUE`, only the regression coefficients will be returned. This is useful when being called from other functions, e.g. `NBDev`. If `coef=FALSE`, a `glm` object will be returned.
`solvers`	The non-linear equation solvers to be used if iterative fitting is necessary.
`verbose`	A logical scalar, indicating whether intermediate messages should be printed.
`maxit`	A positive integer, the maximum number of iterations allowed if iterative fitting is necessary.

`start`	A numeric vector of starting values, with length being the same as the number of columns of `x`.
`infoParms`	A list of three components, named `j`, `k`, and `m`, respectively. These parameters control the type of adjustment made to the score function. When all `j`, `k`, and `m` are 1, the observed information is used in the adjustment. When `k=0`, the expected information is used. See reference for details.
`order`	A positive integer. Usually this should be set to 2, indicating the second order, i.e., $O(n^{-1})$ bias being reduced by the adjustment. For one-way design, if `infoParms$j=1` and offsets are constants within each treatment, the third order bias reduction with `order=3` is also support, resulting in an estimate with both $O(n^{-1})$ order and $O(n^{-2})$ order biases being reduced.
`tol`	Small positive integer, indicating the desired accuracy of parameter estimates.
`qr.tol`	The same as the `tol` argument of qr.

Value

It depends.

Author(s)

Long Qu <rtistician@gmail.com>

Examples

```
## prepare example data
data(mockRNASeqData)
x=mockRNASeqData$design.matrix
y=mockRNASeqData$counts[3462,]
offset = log(mockRNASeqData$estimated.normalization)
overDisp = mockRNASeqData$estimated.nbdisp[3462]
nbfam = negbin('log', overDisp)

## usual maximum likelihood estimate
coef(glm.fit(x, y, offset=offset, family=nbfam))

## 2nd-order biased reduced fit with observed information
ctrl= fbrNBglm.control(infoParms=list(j=1,k=1,m=1), order=2L, coefOnly=TRUE)
fbrNBglm.fit(x, y, offset=offset, family=nbfam, control=ctrl)

## 2nd-order biased reduced fit with expected information
ctrl= fbrNBglm.control(infoParms=list(j=0,k=1,m=1), order=2L, coefOnly=TRUE)
fbrNBglm.fit(x, y, offset=offset, family=nbfam, control=ctrl)

## 3rd-order biased reduced fit with observed information
## Not available yet if offsets are non-constants with a treatment
offset.avg = ave(offset, mockRNASeqData$treatment)
ctrl= fbrNBglm.control(infoParms=list(j=1,k=1,m=1), order=3L, coefOnly=TRUE)
fbrNBglm.fit(x, y, offset=offset.avg, family=nbfam, control=ctrl)
```

`glmSolvers` *Generalized linear model (glm) solvers*

Description

`glmsolve` sequencially tries multiple glm solvers and returns the result from the success. This tries to avoid numerical instabilities that occassionally affect some glm solving routines. If one fails, the next solver will be tried.

Usage

```
glmSolvers

glmsolve(formula, family = gaussian, data, control = list(...),
  solvers = glmSolvers, ...)
```

Arguments

`formula, family, data`	Identical to those of `glm`.
`control`	A list of named control options that specifies the details for each solver. Named elements not used by any solver will be ignored.
`solvers`	A named list of glm solvers, default to `glmSolvers`. Each element is itself a list with components named `solve`, `success`, and `result`, each of which is an expression. `solve` is an expression used to fit the glm model, the result of which is assigned to an object `ans`. `success` is an expression that returns a logical scalar, indicating whether the solver has succeeded. It may refer to the `ans` returned by `solve`. Often, it will check the `converged` element of `ans`, or something similar. `result` is an expression that returns a `glm` object from the success solver. If the `solver` does not return a `glm` object, it is the duty of `result` expression to convert it to a `glm` object.
`...`	Additional arguments passed to solvers.

Format

`glmSolvers` is currently a list of length 4. See `solvers` argument for details and the example.

Details

Currently, the supported are solvers are `c('glm', 'glm.fit3','nlminb', 'BFGS')`. `'glm'` is the one that comes with the default `stats` package. `'glm.fit3'` is a modification with better stability (but slightly slower). `'nlminb'` uses the general optimization routine `nlminb`. `'BFGS'` uses the `method='BFGS'` option provided by `optim`.

Value

If at least one of the solvers succeed, an `glm` object will be returned from the success. If all solvers fail but at least one solver did not throw an error, the result from the last solver that did not throw an error will be returned. In this case, `glmsolve` will throw a warning, stating that none of the solvers succeeded. If all solvers ended up throwing errors, `glmsolve` will also throw an error, again stating that non of the solvers succeeded.

Author(s)

Long Qu

See Also

`glm`, `nlminb`, `optim`, `nlsolve`

Examples

```
str(glmSolvers) ## available solvers stored in glmSolvers object
## Taken from stats::glm:
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glmsolve(counts ~ outcome + treatment, family = poisson())
```

`mockRNASeqData` *A Simulated RNA-Seq Data Set*

Description

This is a simulated RNA-Seq data set using a negative binomial model with 10000 genes and 8 experimental unit, under a balanced two-treatment comparison design.

Usage

```
mockRNASeqData
```

Format

This is a list with the following components:

counts This is a numeric data matrix with 10000 rows and 8 columns, containing counts for each gene (row) and each experimental unit (column).

treatment This is a factor with 2 levels, indicating the treatment group of each column of `counts`.

design.matrix This is an example of design matrix corresponding to `treatment`.

true.normalization This is a numeric vector of normalizing factors actually used to simulate the data matrix.

estimated.normalization This is a numeric vector of normalizing factors estimated from the data matrix, using the so-called "TMM" method.

true.nbdisp This is a numeric vector of negative binomial over-dispersion parameters actually used to simulate the data. This is using the parameterization such that `true.nbdisp = 1/size`, where `size` is the parameter used in `rnbinom`.

estimated.nbdisp This is a numeric vector of estimated negative binomial over-dispersion parameters, using the "TrendedDisp" method from the edgeR package.

ngenes Integer scalar 10000, the number of rows of `counts`.

nsamples Integer scalar 8, the number of columns of `counts`.

true.DEgenes An integer vector of length 3500, indicating the correct row indices of differentially expressed genes, i.e., rows whose means differ across the two treatments.

true.foldChanges A numeric vector of length 3500, indicating the true ratio of means for each differentially expressed genes.

simulation.expression This is a `R` expression that was used to simulate the `mockRNASeqData` data set itself. `eval(mockRNASeqData$simulation.expression)` should generate an identical data set, except for the `simulation.expression` component itself.

`NBDev` *Fit a negative binomial GLM for given design matrix*

Description

A function called within `QL.fit` to fit a negative binomial GLM of each gene for given design matrix

Usage

```
NBDev(counts,design,log.offset,nb.disp,print.progress=TRUE, bias.fold.tolerance=1.10)
```

Arguments

`counts`	RNA-seq data matrix of integer expression counts. Each row contains observations from a single gene. Each column contains observations from a single experimental unit.
`design`	A single element from the `design.list` argument given to `QL.fit`.
`log.offset`	A vector of log-scale, additive factors used to adjust estimated log-scale means for differences in library sizes across samples. Commonly used offsets include,`log.offset=log(colSums(counts))` or `log.offset=log(apply(counts,2,quantile,.75))`. The default setting in `QLfit` makes no adjustment for library sizes (i.e. log.offset=0).
`nb.disp`	estimated negative binomial dispersion parameters obtained from either `estimateGLMTrendedDisp` or `estimateGLMCommonDisp` in package `edgeR`. These estimates are treated as known and are used to compute deviances.
`print.progress`	logical. If `TRUE`, a text progress bar will be displayed during the running of this function.

`bias.fold.tolerance`	A numerical value no smaller than 1. If the bias reduction of maximum likelihood estimates of (log) fold change is likely to result in a ratio of fold changes greater than this value, then bias reduction will be performed on such genes. Setting `bias.fold.tolerance=Inf` will completely disable bias reduction; setting `bias.fold.tolerance=1` will always perform bias reduction. See details.

Details

This functions fits, for each row of `counts`, a negative binomial log-linear model through GLM framework with the over-dispersion parameter fixed. Asymptotic biases of regression coefficients (i.e., log fold changes) are then estimated by a plug-in estimate [eqn. (15.4) of McCullagh and Nelder, 1989] from the last iteration of iteratively re-weighted least squares (IWLS) procedure. The fitted response values are then compared with or without such a bias term. If the ratio of fitted response values are larger than `bias.fold.tolerance` for any observation, then the bias-reduction (not bias-correction) procedure according to Firth (1993) and Kosmidis & Firth (2009) is applied to such rows of `counts`, by adjusting the score equation with a term based on the observed information. Such bias-reduced estimates are more advantageous than directly subtracting the estimated bias from the maximum likelihood estimates as the latter may not exist (e.g., when all counts in one treatment group are zeros).

Value

list containing:

`dev`	vector containing the deviance for each gene under a negative binomial model fit to design matrix specified by `design`. This vector is passed along within the `QL.fit` function.
`means`	matrix of fitted mean values for each gene
`parms`	matrix of estimated coefficients for each gene. Note that these are given on the log scale. (i.e. intercept coefficient reports log(average count) and non-intercept coefficients report estimated (and bias reduced, when appropriate) log fold-changes.)

Author(s)

Steve Lund (<`lundsp@gmail.com`>), Long Qu (<`rtistician@gmail.com`>)

References

Firth (1993) "Bias reduction of maximum likelihood estimates" *Biometrika*, **80**, 27–38.

Kosmidis and Firth (2009) "Bias reduction in exponential family nonlinear models" *Biometrika*, **96**, 793–804.

Lund, Nettleton, McCarthy and Smyth (2012) "Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates" emphSAGMB, **11**(5).

McCullagh and Nelder (1989) "Generalized Linear Models", 2nd edition. London: Chapman and Hall.

Examples

```
## Not run:

## no bias reduction
noReduction =
with(mockRNASeqData,
NBDev(counts, design.matrix, log(estimated.normalization),
  estimated.nbdisp, bias.fold.tolerance=Inf)
)

## bias reduction for all genes
allReduction =
with(mockRNASeqData,
NBDev(counts, design.matrix, log(estimated.normalization),
  estimated.nbdisp, bias.fold.tolerance=1)
)

## default: bias reduction for genes showing large bias
someReduction =
with(mockRNASeqData,
NBDev(counts, design.matrix, log(estimated.normalization),
  estimated.nbdisp, bias.fold.tolerance=1.1)
)


## End(Not run)
```

---

negbin *Negative Binomial Family*

Description

This is an extension of the negbin in the mgcv package, providing additional components related to the negative binomial distribution with log link.

Compared to negbin followed by a call to fix.family.link in the mgcv package this function provides a different implementation of linkinv, added a component initializers of initialization expressions, added the third and fourth order cumulant functions cumulant3 and cumulant4.

Usage

```
negbin(link = "log", overdisp = stop("'overdisp' must be specified"))
```

Arguments

link	The link funciton. Only 'log' is supported currently.
overdisp	A positive overdispersion parameter. This is the same as 1/size, where size is the same as in dnbinom.

Value

A list of class `c("fbrNBfamily", "family")`, containing all components from the `mgcv::negbin` followed by a call to `fix.family.link` with the following additional components:

setTheta A function setting the overdispersion parameter to a specified value. The input argument to this function is `1/overdisp`.

cumulant3 A function returning the third cumulant of the distribution. Input arguments are the mean `mu` and variance `var`.

cumulant4 A function returning the fourth cumulant of the distribution. Input arguments are the mean `mu` and variance `var`.

initializers A list of expressions that can be used to provide starting values for iterative fitting. See the `initialize` component of the result from `family`.

Examples

```
negbin('log', 1)
```

`nlSolvers` *Nonlinear equation solvers*

Description

`nlsolve` sequentially tries multiple nonliear equation solvers until a solution is found or all solvers have been tried.

Usage

```
nlSolvers

nlsolve(start, func, solvers = nlSolvers, control, ...)
```

Arguments

`start`	A numeric vector of starting values.
`func`	An objective function of which the zeros are sought.
`solvers`	A (named) list, with each element being itself a list with components named `solve`, `success` and `result`. See the `solvers` argument of `glmsolve`.
`control`	See the `control` argument of `glmsolve`.
`...`	Additional arguments.

Format

`nlSolvers` is currently a named list of length 8.

Details

Currently supported solvers are `c('broyden', 'nleqslv.Broyden','nleqslv.Newton','fsolve','newtonsys','sane'` They are respectively `broyden` in the `pracma` package, `nleqslv` in the `nleqslv` with `method='Broyden'` or with `method='Newton'`, functions `fsolve` and `newtonsys` in the `pracma` package, and functions `sane`, `dfsane`, and `BBsolve` in the `BB` package.

Value

If at least one solver succeeds, a numeric vector of solutions will be returned, with attribute `attr(*, 'nlsolve.metho')` set to a string indicating the name of the success solver being used, and with attribute `attr(*, 'nlsolve.success')` being a logical scalar indicating whether any solver succeeded. When all solvers fail, the last one is returned, but with `nlsolve.success` being set to `FALSE`.

Author(s)

Long Qu

See Also

`broyden`, `nleqslv`, `fsolve`, `newtonsys`, `sane`, `dfsane`, `BBsolve`, `glmsolve`.

Examples

```
str(nlSolvers) ## list of existing solvers
```

`PoisDev` *Compute Poisson deviances (up to a constant) for given design matrix*

Description

A function called within `QL.fit` to compute Poisson deviances of each gene for given design matrix

Usage

```
PoisDev(counts,design,log.offset,print.progress=TRUE)
```

Arguments

`counts`	RNA-seq data matrix of integer expression counts. Each row contains observations from a single gene. Each column contains observations from a single experimental unit.
`design`	A single element from the `design.list` argument given to `QL.fit`.
`log.offset`	A vector of log-scale, additive factors used to adjust estimated log-scale means for differences in library sizes across samples. Commonly used offsets include,`log.offset=log(colSums(counts))` or `log.offset=log(apply(counts,2,quantile,.75))`. The default setting in `QL.fit` makes no adjustment for library sizes (i.e. log.offset=0).
`print.progress`	logical. If `TRUE`, a text progress bar will be displayed during the running of this function.

Value

list containing:

`dev`	vector containing the deviance for each gene under a Poisson model fit to design matrix (or vector, for one-factor experiments) specified by `design`. This vector is passed along within the `QL.fit` function.
`means`	matrix of fitted mean values for each gene
`parms`	matrix of estimated coefficients for each gene
`bartlett.epsilon`	A numeric vector containing the epsilon values that can be later used for Bartllet correction of likelihood ratio test.

Author(s)

Originally authored by Steve Lund <`lundsp@gmail.com`>; later modified by Long Qu <`rtistician@gmail.com`>

Examples

```
## Not run:

pois.fit =
with(mockRNASeqData,
PoisDev(counts, design.matrix, log(estimated.normalization))
)


## End(Not run)
```

`QL.fit` *Fit quasi-likelihood models to matrix of RNA-seq expression count data*

Description

Fit a quasi-likelihood model to RNA-seq expression count data using the methods detailed in Lund, Nettleton, McCarthy, and Smyth (2012).

Usage

```
QL.fit(counts, design.list, test.mat=cbind(1L, 2:length(design.list)),
   log.offset=NULL, Model="NegBin", print.progress=TRUE, NBdisp="trend",
   bias.fold.tolerance=1.10, ...)
```

Arguments

`counts`	RNA-seq data matrix of integer expression counts. Each row contains observations from a single gene. Each column contains observations from a single experimental unit.
`design.list`	List of design matrices for the full model and reduced model(s). The first element of `design.list` must describe the overall full model, as this design is used to compute deviance residuals for estimating dispersion. One-factor designs may be specified as vectors. The number of rows in each design matrix (or the length of each design vector) must be `ncol(counts)`. Means are modeled with a log link function.
`test.mat`	T by 2 matrix dictating which designs are to be compared, where T is the total number of desired hypothesis tests for each gene. Each row contains two integers, which provide the indices within `design.list` of the two designs to be compared. If `test.mat` is not specified, the default is compare the first design (the full model) to each of the other designs provided in `design.list` in order (i.e. first design compared to second design, first design compared to third design, first design compared to fourth design, etc.). If `length(design.list)=1`, then `test.mat` is ignored.
`log.offset`	A vector of log-scale, additive factors used to adjust estimated log-scale means for differences in library sizes across samples. Commonly used offsets include `log.offset=log(colSums(counts))` and `log.offset=log(apply(counts[rowSums(counts)!=0,],` The default setting makes no adjustment for library sizes (i.e. log.offset=0).
`Model`	Must be one of "Poisson" or "NegBin", specifying use of a quasi-Poisson or quasi-negative binomial model, respectively.
`print.progress`	logical. If `TRUE`, a text progress bar will be displayed during the fitting of *each* models in the `design.list`.
`NBdisp`	Used only when `Model="NegBin"`. Must be one of "trend", "common" or a vector of non-negative real numbers with length equal to `nrow(counts)`. Specifying `NBdisp="trend"` or `NBdisp="common"` will use estimateGLMTrendedDisp or `estimateGLMCommonDisp`, respectively, from the package `edgeR` to estimate negative binomial dispersion parameters for each gene. Estimates obtained from other sources can be used by entering `NBdisp` as a vector containing the negative binomial dispersion value to use for each gene when fitting quasi-likelihood model.
`bias.fold.tolerance`	A numerical value no smaller than 1. If the bias reduction of maximum likelihood estimates of (log) fold change is likely to result in a ratio of fold changes greater than this value, then bias reduction will be performed on such genes. Setting `bias.fold.tolerance=Inf` will completely disable bias reduction; setting `bias.fold.tolerance=1` will always perform bias reduction. See `NBDev` for details. Currently, this option is ignored unless `Model="NegBin"`.
`...`	arguments to be passed to the function `estimateGLMTrendedDisp` or `estimateGLMCommonDisp` from the package `edgeR`.

Value

list containing:

`"LRT"`	matrix providing unadjusted likelihood ratio test statistics. Each column contains statistics from a single hypothesis test, applied separately to each gene.
`"LRT.Bart"`	The same as `LRT`, except that the LRT test statistics are adjusted by the Bartlett factor to better match their asymptotic moments.
`"phi.hat.dev"`	vector providing unshrunken, deviance-based estimates of quasi-dispersion (phi) for each gene.
`"phi.hat.pearson"`	vector providing unshrunken, Pearson estimates of quasi-dispersion (phi) for each gene.
`"mn.cnt"`	vector providing average count for each gene.
`"den.df"`	denominator degrees of freedom. Equal to the number of samples minus the number of fitted parameters in the full model, which is specified by the first element of `design.list`
`"num.df"`	vector of numerator degrees of freedom for each test, computed as the difference in the number of fitted parameters between the full and reduced models for each test.
`"Model"`	Either "Poisson" or "NegBin", specifying which model (quasi-Poisson or quasi-negative binomial, respectively) was used.
`"nb.disp"`	Only appears when `Model="NegBin"`. Vector providing negative binomial dispersion parameter estimate used during fitting of quasi-negative binomial model for each gene.
`fitted.values`	matrix of fitted mean values
`coefficients`	matrix of estimated coefficients for each gene. Note that these are given on the log scale. (i.e. intercept coefficient reports log(average count) and non-intercept coefficients report estimated (and bias reduced, when appropriate) log fold-changes.)

Author(s)

Originally authored by Steve Lund <`lundsp@gmail.com`>; later modified by Long Qu <`rtistician@gmail.com`>

References

Kosmidis and Firth (2009) "Bias reduction in exponential family nonlinear models" *Biometrika*, **96**, 793–804.

Lund, Nettleton, McCarthy and Smyth (2012) "Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates" *SAGMB*, **11**(5).

McCarthy, Chen and Smyth (2012) "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation" *Nucleic Acids Res.* **40**(10), 4288–97.

Cordeiro (1983) "Improved Likelihood Ratio Statistics for Generalized Linear Models" *Journal of the Royal Statistical Society. Series B (Methodological)*, **45**, 404–413.

See Also

`QL.results`, `NBDev`, `PoisDev`, `mockRNASeqData`

Examples

```
data(mockRNASeqData)

### Create list of designs describing model under null and alternative hypotheses
design.list=list(
mockRNASeqData$design.matrix,
matrix(1, nrow=rep(mockRNASeqData$nsamples))
)

## Not run:
### Analyze using QL, QLShrink and QLSpline methods applied to quasi-Poisson model
fit =
with(mockRNASeqData,
QL.fit(counts, design.list,
log.offset=log(estimated.normalization),
 Model="Poisson", bias.fold.tolerance=Inf)
)
results = QL.results(fit)

### How many significant genes at FDR=.05 from QLSpline method?
apply(results$Q.values[[3]]<.05,2,sum)

### Indexes for Top 10 most significant genes from QLSpline method
head(order(results$P.values[[3]]), 10)

## End(Not run)

## Not run:
### Analyze using QL, QLShrink and QLSpline methods
### applied to quasi-negative binomial model
fit2 =
with(mockRNASeqData,
QL.fit(counts, design.list,
log.offset=log(estimated.normalization),
Model="NegBin")
)
##########################################################
## Note: 'NBdisp' typically will not be specified when  ##
## calling QL.fit while analyzing real data. Providing  ##
## numeric values for 'NBdisp' prevents neg binomial    ##
## dispersions from being estimated from the data.      ##
##########################################################
results2<-QL.results(fit2)

### How many significant genes at FDR=.05 for QLSpline method?
apply(results2$Q.values[[3]]<.05,2,sum)

### Indexes for Top 10 most significant genes from QLShrink method
head(order(results2$P.values[[2]]), 10)
```

```
## End(Not run)
```

`QL.results` *Obtain p-values and q-values using results from* `QL.fit`

Description

Obtain significance results for quasi-likelihood model fits to RNA-seq expression count data using the methods detailed in Lund, Nettleton, McCarthy, and Smyth (2012).

Usage

```
QL.results(fit,Dispersion="Deviance",spline.df=NULL,Plot=TRUE)
```

Arguments

`fit`	The list returned by the function QL.fit
`Dispersion`	Must be one of "Deviance" or "Pearson", specifying which type of estimator should be used for estimating quasi-likelihood dispersion parameter.
`spline.df`	Optional. User may specify the degrees of freedom to use when fitting a cubic spline to log-scale(estimated dispersion) versus the log(average count). Default uses cross-validation in `sreg` function to pick appropriate degrees of freedom.
`Plot`	logical. If TRUE, the estimated dispersion versus the average count are plotted on a log-scale with the corresponding cubic spline fit overlaid.

Value

list containing:

`"P.values"`	list of matrices providing p-values for the QL, QLShrink and QLSpline methods, respectively. The i^th column of each element of `pvals` corresponds to the hypothesis test assigned in the i^th row of `test.mat`.
`"Q.values"`	list of matrices providing q-values for the QL, QLShrink and QLSpline methods, respectively. The i^th column of each element of `qvals` corresponds to the hypothesis test assigned in the i^th row of `test.mat`. Q-values are computed using the methods of Nettleton et al. (2006) JABES 11, 337-356.
`"F.stat"`	list of matrices providing F-statistics for the QL, QLShrink and QLSpline methods, respectively. The i^th column of each element of `F.stat` corresponds to the hypothesis test assigned in the i^th row of `test.mat`.
`"m0"`	matrix providing estimated number of true null hypotheses for each test(arranged by row) under each of the three methods(arranged by column). `m0` values are computed using the methods of Nettleton et al. (2006) JABES 11, 337-356.
`"d0"`	vector containing estimated additional denominator degrees of freedom gained from shrinking dispersion estimates in the QLShrink and QLSpline procedures, respectively.

Author(s)

Steve Lund <lundsp@gmail.com>

References

Lund, Nettleton, McCarthy and Smyth (2012) "Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates" *SAGMB*, **11**(5).

See Also

QL.fit, NBDev, mockRNASeqData

Examples

```
## see examples for QL.fit()
```

Index

∗ **Firth bias reduction**
    fbrNBglm, 3
∗ **Poisson**
    PoisDev, 11
∗ **RNA-seq**
    NBDev, 7
    PoisDev, 11
    QL.fit, 12
    QL.results, 16
∗ **bias correction**
    coef.glm, 2
∗ **datasets**
    glmSolvers, 5
    mockRNASeqData, 6
    nlSolvers, 10
∗ **differential expression**
    NBDev, 7
    PoisDev, 11
    QL.fit, 12
    QL.results, 16
∗ **distribution**
    NBDev, 7
    PoisDev, 11
∗ **htest**
    QL.results, 16
∗ **iteration**
    fbrNBglm, 3
    glmSolvers, 5
    NBDev, 7
    nlSolvers, 10
    PoisDev, 11
    QL.fit, 12
∗ **methods**
    coef.glm, 2
∗ **models**
    fbrNBglm, 3
    QL.fit, 12
∗ **multivariate**
    QL.fit, 12
∗ **negative binomial**
    NBDev, 7
    QL.fit, 12
    QL.results, 16
∗ **poisson**
    QL.fit, 12
    QL.results, 16
∗ **quasi-likelihood**
    NBDev, 7
    PoisDev, 11
    QL.fit, 12
    QL.results, 16
∗ **regression**
    fbrNBglm, 3
    QL.fit, 12
    QL.results, 16
∗
    coef.glm, 2

BBsolve, *11*
broyden, *11*

coef.glm, 2

dfsane, *11*
dnbinom, *3*, *9*

edgeR, *7*
estimateGLMCommonDisp, *13*
estimateGLMTrendedDisp, *13*

family, *10*
fbrNBglm, 3
fix.family.link, *9*, *10*
fsolve, *11*

glm, *2*, *5*, *6*
glm.fit, *3*
glmSolve (glmSolvers), 5
glmsolve, *10*, *11*
glmsolve (glmSolvers), 5

glmSolvers, 5

mockRNASeqData, 6, *14*, *17*

NBDev, *3*, 7, *13*, *14*, *17*
negbin, *9*, 9, *10*
newtonsys, *11*
nleqslv, *11*
nlminb, *5*, *6*
nlSolve *(*nlSolvers*)*, 10
nlsolve, *6*
nlsolve *(*nlSolvers*)*, 10
nlSolvers, 10

optim, *5*, *6*

PoisDev, 11, *14*

QL.fit, 12, *17*
QL.results, *14*, 16
qr, *4*

rnbinom, *7*

sane, *11*