Package 'edge'

November 11, 2026

Type Package

Title Extraction of Differential Gene Expression

Date 2015-04-15

Version 2.44.0

Author John D. Storey, Jeffrey T. Leek and Andrew J. Bass

Maintainer John D. Storey <jstorey@princeton.edu>, Andrew J. Bass <ajbass@emory.edu>

biocViews MultipleComparison, DifferentialExpression, TimeCourse, Regression, GeneExpression, DataImport

Description The edge package implements methods for carrying out differential expression analyses of genome-wide gene expression studies. Significance testing using the optimal discovery procedure and generalized likelihood ratio tests (equivalent to F-tests and t-tests) are implemented for general study designs. Special functions are available to facilitate the analysis of common study designs, including time course experiments. Other packages such as sva and qvalue are integrated in edge to provide a wide range of tools for gene expression analysis.

VignetteBuilder knitr

Imports methods, splines, sva, qvalue(>= 1.99.0), MASS

Suggests testthat, knitr, ggplot2, reshape2

Depends R(>= 3.1.0), Biobase

URL https://github.com/jdstorey/edge

BugReports https://github.com/jdstorey/edge/issues

LazyData true

License MIT + file LICENSE

NeedsCompilation yes

RoxygenNote 5.0.1

git_url https://git.bioconductor.org/packages/edge

git_branch RELEASE_3_23

git_last_commit 7c0ffa4

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

apply_qvalue . . . 3
apply_sva . . . 4
betaCoef . . . 5
build_models . . . 6
build_study . . . 7
deFit-class . . . 9
deSet . . . 10
deSet-class . . . 11
edge . . . 12
endotoxin . . . 13
fitFull . . . 14
fitNull . . . 15
fit_models . . . 16
fullMatrix . . . 18
fullModel . . . 19
gibson . . . 20
individual . . . 22
kidney . . . 23
kl_clust . . . 24
lrt . . . 26
nullMatrix . . . 28
nullModel . . . 29
odp . . . 31
qvalueObj . . . 33
resFull . . . 34
resNull . . . 35
show . . . 36
sType . . . 37
summary . . . 38

Index **40**

apply_qvalue *Estimate the q-values for a given set of p-values*

Description

Runs qvalue on a deSet object.

Usage

```
apply_qvalue(object, ...)

## S4 method for signature 'deSet'
apply_qvalue(object, ...)
```

Arguments

object	S4 object: deSet
...	Additional arguments for qvalue

Value

deSet object with slots updated by qvalue calculations.

Author(s)

John Storey, Andrew Bass

References

Storey JD and Tibshirani R. (2003) Statistical significance for genome-wide studies. Proceedings of the National Academy of Sciences, 100: 9440-9445

See Also

deSet, odp and lrt

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)
```

```
# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# Run lrt (or odp) and apply_qvalue
de_lrt <- lrt(de_obj)
de_lrt <- apply_qvalue(de_lrt, fdr.level = 0.05,
pi0.method = "bootstrap", adj=1.2)
summary(de_lrt)
```

`apply_sva` *Estimate surrogate variables*

Description

Runs `sva` on the null and full models in `deSet`. See `sva` for additional details.

Usage

```
apply_sva(object, ...)

## S4 method for signature 'deSet'
apply_sva(object, ...)
```

Arguments

`object`	S4 object: `deSet`
`...`	Additional arguments for `sva`

Value

`deSet` object where the surrogate variables estimated by `sva` are added to the full model and null model matrices.

Author(s)

John Storey, Jeffrey Leek, Andrew Bass

References

Leek JT, Storey JD (2007) Capturing Heterogeneity in Gene Expression Studies by Surrogate Variable Analysis. PLoS Genet 3(9): e161. doi:10.1371/journal.pgen.0030161

Leek JT and Storey JD. (2008) A general framework for multiple testing dependence. Proceedings of the National Academy of Sciences, 105: 18718- 18723.

See Also

deSet, odp and lrt

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run surrogate variable analysis
de_sva <- apply_sva(de_obj)

# run odp/lrt with surrogate variables added
de_odp <- odp(de_sva, bs.its = 30)
summary(de_odp)
```

---

betaCoef *Regression coefficients from full model fit*

Description

Access the full model fitted coefficients of a deFit object.

Usage

```
betaCoef(object)

## S4 method for signature 'deFit'
betaCoef(object)
```

Arguments

object	S4 object: deFit

Value

betaCoef returns the regression coefficients for the full model fit.

Author(s)

John Storey, Andrew Bass

See Also

fit_models

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run fit_models to get model fits
de_fit <- fit_models(de_obj)

# extract beta coefficients
beta <- betaCoef(de_fit)
```

---

build_models *Generate a deSet object with full and null models*

Description

build_models creates a deSet object. The user inputs the full and null models.

Usage

```
build_models(data, cov, full.model = NULL, null.model = NULL, ind = NULL)
```

Arguments

data	matrix: gene expression data.
cov	data.frame: the covariates in the study.
full.model	formula: the adjustment and the biological variables of interest.
null.model	formula: the adjustment variables.
ind	factor: individuals sampled in the study. Default is NULL. Optional.

Value

deSet object

Author(s)

John Storey, Andy Bass

See Also

deSet, build_study

Examples

```
# create ExpressionSet object from kidney dataset
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null.model <- ~sex
full.model <- ~sex + ns(age, df=4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null.model,
full.model = full.model)
```

---

build_study *Formulates the experimental models*

Description

build_study generates the full and null models for users unfamiliar with building models in R. There are two types of experimental designs: static and time-course. For more details, refer to the vignette.

Usage

```
build_study(data, grp = NULL, adj.var = NULL, bio.var = NULL,
  tme = NULL, ind = NULL, sampling = c("static", "timecourse"),
  basis.df = 2, basis.type = c("ncs", "poly"))
```

Arguments

data	matrix: gene expression data (rows are genes, columns are samples).
grp	vector: group assignement in the study (for K-class studies). Optional.
adj.var	matrix: adjustment variables. Optional.
bio.var	matrix: biological variables. Optional.
tme	vector: time variable in a time course study. Optional.
ind	factor: individual factor for repeated observations of the same individuals. Optional.
sampling	string: type of study. Either "static" or "timecourse". Default is "static".
basis.df	numeric: degrees of freedom of the basis for time course study. Default is 2.
basis.type	string: either "ncs" (natural cubic spline) or "ps" (polynomial spline) basis for time course study. Default is "ncs".

Value

deSet object

Author(s)

John Storey, Andy Bass

See Also

deSet, build_models

Examples

```
# create ExpressionSet object from kidney dataset
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr

# create deSet object from data
de_obj <- build_study(data = kidexpr, adj.var = sex, tme = age,
sampling = "timecourse", basis.df = 4)
```

deFit-class *The differential expression class for the model fits*

Description

Object returned from `fit_models` containing information regarding the model fits for the experiment.

Slots

`fit.full matrix`: containing fitted values for the full model.

`fit.null matrix`: containing fitted values for the null model.

`res.full matrix`: the residuals of the full model.

`res.null matrix`: the residuals of the null model.

`dH.full vector`: contains diagonal elements in the projection matrix for the full model.

`beta.coef matrix`: fitted coefficients for the full model.

`stat.type string`: information on the statistic of interest. Currently, the only options are "lrt" and "odp".

Methods

`fitNull(deFit)` Access fitted data from null model.

`fitFull(deFit)` Access fitted data from full model.

`resNull(deFit)` Access residuals from null model fit.

`resFull(deFit)` Access residuals from full model fit.

`betaCoef(deFit)` Access beta coefficients in linear model.

`sType(deFit)` Access statistic type of model fitting utilized in function.

Author(s)

John Storey, Jeffrey Leek, Andrew Bass

deSet *Create a deSet object from an ExpressionSet*

Description

Creates a deSet object that extends the ExpressionSet object.

Usage

```
deSet(object, full.model, null.model, individual = NULL)

## S4 method for signature 'ExpressionSet'
deSet(object, full.model, null.model,
  individual = NULL)
```

Arguments

object	S4 object: ExpressionSet
full.model	formula: full model containing the both the adjustment and the biological variables for the experiment.
null.model	formula: null model containing the adjustment variables for the experiment.
individual	factor: information on repeated samples in experiment.

Value

deSet object

Note

It is essential that the null and full models have the same variables as the ExpressionSet phenoType column names.

Author(s)

John Storey, Andrew Bass

See Also

deSet, odp and lrt

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
```

```
cov <- data.frame(sex = sex, age = age)
pDat <- as(cov, "AnnotatedDataFrame")
exp_set <- ExpressionSet(assayData = kidexpr, phenoData = pDat)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- deSet(exp_set, null.model = null_model,
full.model = full_model)

# optionally add individuals to experiment, in this case there are 36
# individuals that were sampled twice
indSamples <- as.factor(rep(1:36, each = 2))
de_obj <- deSet(exp_set, null.model = null_model,
full.model = full_model, ind = indSamples)
summary(de_obj)
```

`deSet-class` *The differential expression class (deSet)*

Description

The `deSet` class extends the `ExpressionSet` class. While the `ExpressionSet` class contains information about the experiment, the `deSet` class contains both experimental information and additional information relevant for differential expression analysis, explained below in Slots.

Slots

`null.model formula`: contains the adjustment variables in the experiment. The null model is used for comparison when fitting the full model.

`full.model formula`: contains the adjustment variables and the biological variables of interest.

`null.matrix matrix`: the null model as a matrix.

`full.matrix matrix`: the full model as a matrix.

`individual factor`: contains information on which sample is from which individual in the experiment.

`qvalueObj S3 object`: containing `qvalue` object. See `qvalue` for additional details.

Methods

`as(ExpressionSet, "deSet")` Coerce objects of `ExpressionSet` to `deSet`.

`lrt(deSet, ...)` Performs a generalized likelihood ratio test using the full and null models.

`odp(deSet, ...)` Performs the optimal discovery procedure, which is a new approach for optimally performing many hypothesis tests in a high-dimensional study.

`kl_clust(deSet, ...)` An implementation of mODP that assigns genes to modules based off of the Kullback-Leibler distance.

`fit_models(deSet, ...)` Fits a linear model to each gene by method of least squares.

`apply_qvalue(deSet, ...)` Applies `qvalue` function.

`apply_sva(deSet, ...)` Applies surrogate variable analysis (`sva`).

`fullMatrix(deSet)` Access and set full matrix.

`nullMatrix(deSet)` Access and set null matrix.

`fullModel(deSet)` Access and set full model.

`nullModel(deSet)` Access and set null model.

`individual(deSet)` Access and set individual slot.

`qvalueObj(deSet)` Access `qvalue` object. See `qvalue`.

`validObject(deSet)` Check validity of `deSet` object.

Note

See `ExpressionSet` for additional slot information.

Author(s)

John Storey, Jeffrey Leek, Andrew Bass

---

edge *Extraction of Differential Gene Expression*

Description

The edge package implements methods for carrying out differential expression analyses of genome-wide gene expression studies. Significance testing using the optimal discovery procedure and generalized likelihood ratio tests (equivalent to F-tests and t-tests) are implemented for general study designs. Special functions are available to facilitate the analysis of common study designs, including time course experiments. Other packages such as snm, sva, and qvalue are integrated in edge to provide a wide range of tools for gene expression analysis.

Author(s)

John Storey, Jeffrey Leek, Andrew Bass

Examples

```
## Not run:
browseVignettes("edge")

## End(Not run)
```

endotoxin *Gene expression dataset from Calvano et al. (2005) Nature*

Description

The data provide gene expression measurements in an endotoxin study where four subjects were given endotoxin and four subjects were given a placebo. Blood samples were collected and leukocytes were isolated from the samples before infusion and at times 2, 4, 6, 9, 24 hours.

Usage

```
data(endotoxin)
```

Format

- endoexpr: A 500 rows by 46 columns data frame containing expression values.
- class: A vector of length 46 containing information about which individuals were given endotoxin.
- ind: A vector of length 46 providing indexing measurements for each individual in the experiment.
- time: A vector of length 46 indicating time measurements.

Value

endotoxin dataset

Note

The data is a random subset of 500 genes from the full dataset. To download the full data set, go to http://genomine.org/edge/.

References

Storey JD, Xiao W, Leek JT, Tompkins RG, and Davis RW. (2005) Significance analysis of time course microarray experiments. PNAS, 102: 12837-12842.
http://www.pnas.org/content/100/16/9440.full

Examples

```
library(splines)
# import data
data(endotoxin)
ind <- endotoxin$ind
class <- endotoxin$class
time <- endotoxin$time
endoexpr <- endotoxin$endoexpr
cov <- data.frame(individual = ind, time = time, class = class)
```

```
# formulate null and full models in experiement
# note: interaction term is a way of taking into account group effects
mNull <- ~ns(time, df=4, intercept = FALSE) + class
mFull <- ~ns(time, df=4, intercept = FALSE) +
          ns(time, df=4, intercept = FALSE):class + class

# create deSet object
de_obj <- build_models(endoexpr, cov = cov, full.model = mFull,
                       null.model = mNull, ind = ind)

# Perform ODP/lrt statistic to determine significant genes in study
de_odp <- odp(de_obj, bs.its = 10)
de_lrt <- lrt(de_obj, nullDistn = "bootstrap", bs.its = 10)

# summarize significance results
summary(de_odp)
```

`fitFull` *Fitted data from the full model*

Description

Access the fitted data from the full model in a `deFit` object.

Usage

```
fitFull(object)

## S4 method for signature 'deFit'
fitFull(object)
```

Arguments

`object`	`S4 object: deFit`

Value

`fitFull` returns a matrix of fitted values from full model.

Author(s)

John Storey, Andrew Bass

See Also

`fit_models`

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run fit_models to get model fits
de_fit <- fit_models(de_obj)

# extract fitted values for full model
fitted_full <- fitFull(de_fit)
```

`fitNull` *Fitted data from the null model*

Description

Access the fitted data from the null model in an `deFit` object.

Usage

```
fitNull(object)

## S4 method for signature 'deFit'
fitNull(object)
```

Arguments

`object`	`S4 object: deFit`

Value

`fitNull` returns a matrix of fitted values from null model.

Author(s)

John Storey, Andrew Bass

See Also

fit_models

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run fit_models to get model fits
de_fit <- fit_models(de_obj)

# extract fitted values from null model
fitted_null <- fitNull(de_fit)
```

---

fit_models *Linear regression of the null and full models*

Description

fit_models fits a model matrix to each gene by using the least squares method. Model fits can be either statistic type "odp" (optimal discovery procedure) or "lrt" (likelihood ratio test).

Usage

```
fit_models(object, stat.type = c("lrt", "odp"), weights = NULL)

## S4 method for signature 'deSet'
fit_models(object, stat.type = c("lrt", "odp"),
  weights = NULL)
```

Arguments

object	S4 object: deSet.
stat.type	character: type of statistic to be used. Either "lrt" or "odp". Default is "lrt".
weights	matrix: weights for each observation. Default is NULL.

Details

If "odp" method is implemented then the null model is removed from the full model (see Storey 2007). Otherwise, the statistic type has no affect on the model fit.

Value

deFit object

Note

fit_models does not have to be called by the user to use odp, lrt or kl_clust as it is an optional input and is implemented in the methods. The deFit object can be created by the user if a different statistical implementation is required.

Author(s)

John Storey

References

Storey JD. (2007) The optimal discovery procedure: A new approach to simultaneous significance testing. Journal of the Royal Statistical Society, Series B, 69: 347-368.

Storey JD, Dai JY, and Leek JT. (2007) The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments. Biostatistics, 8: 414-432.

Storey JD, Xiao W, Leek JT, Tompkins RG, and Davis RW. (2005) Significance analysis of time course microarray experiments. Proceedings of the National Academy of Sciences, 102: 12837-12842.

See Also

deFit, odp and lrt

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)
```

```
# retrieve statistics from linear regression for each gene
fit_lrt <- fit_models(de_obj, stat.type = "lrt") # lrt method
fit_odp <- fit_models(de_obj, stat.type = "odp") # odp method

# summarize object
summary(fit_odp)
```

`fullMatrix` *Matrix representation of full model*

Description

These generic functions access and set the full matrix for `deSet` object.

Usage

```
fullMatrix(object)

fullMatrix(object) <- value

## S4 method for signature 'deSet'
fullMatrix(object)

## S4 replacement method for signature 'deSet'
fullMatrix(object) <- value
```

Arguments

`object`	`S4 object: deSet`
`value`	`matrix`: full model matrix where the columns are the covariates and rows are observations

Value

`fullMatrix` returns the value of the full model matrix.

Author(s)

Andrew Bass, John Storey

See Also

`deSet`, `fullModel`

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# extract the full model equation as a matrix
mat_full <- fullMatrix(de_obj)
```

`fullModel` *Full model equation*

Description

These generic functions access and set the full model for `deSet` object.

Usage

```
fullModel(object)

fullModel(object) <- value

## S4 method for signature 'deSet'
fullModel(object)

## S4 replacement method for signature 'deSet'
fullModel(object) <- value
```

Arguments

`object`	`S4 object: deSet`
`value`	`formula`: The experiment design for the full model.

Value

the formula for the full model.

Author(s)

John Storey, Andrew Bass

See Also

deSet

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# extract out the full model equation
mod_full <- fullModel(de_obj)

# change the full model in the experiment
fullModel(de_obj) <- ~sex + ns(age, df = 2)
```

---

gibson *Gene expression dataset from Idaghdour et al. (2008)*

Description

The data provide gene expression measurements in peripheral blood leukocyte samples from three Moroccan groups leading distinct ways of life: desert nomadic (DESERT), mountain agrarian (VILLAGE), and coastal urban (AGADIR).

Usage

```
data(gibson)
```

Format

- batch: Batches in experiment.
- location: Environment/lifestyle of Moroccan Amazigh groups.
- gender: Sex of individuals.
- gibexpr: A 500 rows by 46 columns matrix of gene expression values.

Value

gibson dataset

Note

These data are a random subset of 500 genes from the total number of genes in the original data set. To download the full data set, go to http://genomine.org/de/.

References

Idaghdour Y, Storey JD, Jadallah S, and Gibson G. (2008) A genome-wide gene expression signature of lifestyle in peripheral blood of Moroccan Amazighs. PLoS Genetics, 4: e1000052.

Examples

```
# import
data(gibson)
batch <- gibson$batch
gender <- gibson$gender
location <- gibson$location
gibexpr <- gibson$gibexpr
cov <- data.frame(Batch = batch, Gender = gender,
Location = location)

# create deSet for experiment- static experiment
mNull <- ~Gender + Batch
mFull <- ~Gender + Batch + Location

# create deSet object
de_obj <- build_models(gibexpr, cov = cov, full.model = mFull,
null.model = mNull)

# Perform ODP/lrt statistic to determine significant genes in study
de_odp <- odp(de_obj, bs.its = 10)
de_lrt <- lrt(de_obj, nullDistn = "bootstrap", bs.its = 10)

# summarize significance results
summary(de_odp)
```

individual *Individuals sampled in experiment*

Description

These generic functions access and set the individual slot in deSet.

Usage

```
individual(object)

individual(object) <- value

## S4 method for signature 'deSet'
individual(object)

## S4 replacement method for signature 'deSet'
individual(object) <- value
```

Arguments

object	deSet
value	factor: Identifies which samples correspond to which individuals. Important if the same individuals are sampled multiple times in a longitudinal fashion.

Value

individual returns information regarding dinstinct individuals sampled in the experiment.

Author(s)

John Storey, Andrew Bass

See Also

deSet

Examples

```
library(splines)
# import data
data(endotoxin)
ind <- endotoxin$ind
time <- endotoxin$time
class <- endotoxin$class
endoexpr <- endotoxin$endoexpr
cov <- data.frame(individual = ind, time = time, class = class)
```

```
# create ExpressionSet object
pDat <- as(cov, "AnnotatedDataFrame")
exp_set <- ExpressionSet(assayData = endoexpr, phenoData = pDat)

# formulate null and full models in experiement
# note: interaction term is a way of taking into account group effects
mNull <- ~ns(time, df=4, intercept = FALSE)
mFull <- ~ns(time, df=4, intercept = FALSE) +
ns(time, df=4, intercept = FALSE):class + class

# create deSet object
de_obj <- deSet(exp_set, full.model = mFull, null.model = mNull,
individual = ind)

# extract out the individuals factor
ind_exp <- individual(de_obj)
```

`kidney` *Gene expression dataset from Rodwell et al. (2004)*

Description

Gene expression measurements from kidney samples were obtained from 72 human subjects ranging in age from 27 to 92 years. Only one array was obtained per individual, and the age and sex of each individual were recorded.

Usage

```
data(kidney)
```

Format

- kidcov: A 133 rows by 6 columns data frame detailing the study design.
- kidexpr: A 500 rows by 133 columns matrix of gene expression values, where each row corresponds to a different probe-set and each column to a different tissue sample.
- age: A vector of length 133 giving the age of each sample.
- sex: A vector of length 133 giving the sex of each sample.

Value

kidney dataset

Note

These data are a random subset of 500 probe-sets from the total number of probe-sets in the original data set. To download the full data set, go to http://genomine.org/edge/. The `age` and `sex` are contained in `kidcov` data frame.

References

Storey JD, Xiao W, Leek JT, Tompkins RG, and Davis RW. (2005) Significance analysis of time course microarray experiments. PNAS, 102: 12837-12842.
http://www.pnas.org/content/100/16/9440.full

Examples

```
# import data
data(kidney)
sex <- kidney$sex
age <- kidney$age
kidexpr <- kidney$kidexpr

# create model
de_obj <- build_study(data = kidexpr, adj.var = sex, tme = age,
sampling = "timecourse", basis.df = 4)

# use the ODP/lrt method to determine significant genes
de_odp <- odp(de_obj, bs.its=10)
de_lrt <- lrt(de_obj, nullDistn = "bootstrap", bs.its = 10)

# summarize significance results
summary(de_odp)
```

---

kl_clust *Modular optimal discovery procedure (mODP)*

Description

kl_clust is an implementation of mODP that assigns genes to modules based on of the Kullback-Leibler distance.

Usage

```
kl_clust(object, de.fit = NULL, n.mods = 50)

## S4 method for signature 'deSet,missing'
kl_clust(object, de.fit = NULL, n.mods = 50)

## S4 method for signature 'deSet,deFit'
kl_clust(object, de.fit = NULL, n.mods = 50)
```

Arguments

object	S4 object: deSet.
de.fit	S4 object: deFit.
n.mods	integer: number of modules (i.e., clusters).

Details

mODP utilizes a k-means clustering algorithm where genes are assigned to a cluster based on the Kullback-Leiber distance. Each gene is assigned an module-average parameter to calculate the ODP score (See Woo, Leek and Storey 2010 for more details). The mODP and full ODP produce nearly exact results but mODP has the advantage of being computationally faster.

Value

A list with the following slots:

- mu.full: cluster averaged fitted values from full model.
- mu.null: cluster averaged fitted values from null model.
- sig.full: cluster standard deviations from full model.
- sig.null: cluster standard deviations from null model.
- n.per.mod: total members in each cluster.
- clustMembers: cluster membership for each gene.

Note

The results are generally insensitive to the number of modules after a certain threshold of about n.mods>=50 in our experience. It is recommended that users experiment with the number of modules. If the number of modules is equal to the number of genes then the original ODP is implemented.

Author(s)

John Storey, Jeffrey Leek

References

Storey JD. (2007) The optimal discovery procedure: A new approach to simultaneous significance testing. Journal of the Royal Statistical Society, Series B, 69: 347-368.

Storey JD, Dai JY, and Leek JT. (2007) The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments. Biostatistics, 8: 414-432.

Woo S, Leek JT, Storey JD (2010) A computationally efficient modular optimal discovery procedure. Bioinformatics, 27(4): 509-515.

See Also

odp, fit_models

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
```

```
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# mODP method
de_clust <- kl_clust(de_obj)

# change the number of clusters
de_clust <- kl_clust(de_obj, n.mods = 10)

# input a deFit object
de_fit <- fit_models(de_obj, stat.type = "odp")
de_clust <- kl_clust(de_obj, de.fit = de_fit)
```

`lrt` *Performs F-test (likelihood ratio test using Normal likelihood)*

Description

`lrt` performs a generalized likelihood ratio test using the full and null models.

Usage

```
lrt(object, de.fit, nullDistn = c("normal", "bootstrap"), weights = NULL,
  bs.its = 100, seed = NULL, verbose = TRUE, mod.F = FALSE, ...)

## S4 method for signature 'deSet,missing'
lrt(object, de.fit, nullDistn = c("normal",
  "bootstrap"), weights = NULL, bs.its = 100, seed = NULL,
  verbose = TRUE, mod.F = FALSE, ...)

## S4 method for signature 'deSet,deFit'
lrt(object, de.fit, nullDistn = c("normal",
  "bootstrap"), weights = NULL, bs.its = 100, seed = NULL,
  verbose = TRUE, mod.F = FALSE, ...)
```

Arguments

`object`	S4 object: deSet.
`de.fit`	S4 object: deFit. Optional.

nullDistn	character: either "normal" or "bootstrap", If "normal" then the p-values are calculated using the F distribution. If "bootstrap" then a bootstrap algorithm is implemented to simulate statistics from the null distribution. In the "bootstrap" case, empirical p-values are calculated using the observed and null statistics (see empPvals). Default is "normal".
weights	matrix: weights for each observation. Default is NULL.
bs.its	integer: number of null statistics generated (only applicable for "bootstrap" method). Default is 100.
seed	integer: set the seed value. Default is NULL.
verbose	boolean: print iterations for bootstrap method. Default is TRUE.
mod.F	boolean: Moderated F-test, recommended for experiments with a small sample size. Default is FALSE.
...	Additional arguments for apply_qvalue and empPvals function.

Details

lrt fits the full and null models to each gene using the function fit_models and then performs a likelihood ratio test. The user has the option to calculate p-values a Normal distribution assumption or through a bootstrap algorithm. If nullDistn is "bootstrap" then empirical p-values will be determined from the qvalue package (see empPvals).

Value

deSet object

Author(s)

John Storey, Andrew Bass

References

Storey JD, Xiao W, Leek JT, Tompkins RG, and Davis RW. (2005) Significance analysis of time course microarray experiments. Proceedings of the National Academy of Sciences, 102: 12837-12842.

http://en.wikipedia.org/wiki/Likelihood-ratio_test

See Also

deSet, build_models, odp

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
```

```
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# lrt method
de_lrt <- lrt(de_obj, nullDistn = "normal")

# to generate p-values from bootstrap
de_lrt <- lrt(de_obj, nullDistn = "bootstrap", bs.its = 30)

# input a deFit object directly
de_fit <- fit_models(de_obj, stat.type = "lrt")
de_lrt <- lrt(de_obj, de.fit = de_fit)

# summarize object
summary(de_lrt)
```

`nullMatrix` *Matrix representation of null model*

Description

These generic functions access and set the null matrix for `deSet` object.

Usage

```
nullMatrix(object)

nullMatrix(object) <- value

## S4 method for signature 'deSet'
nullMatrix(object)

## S4 replacement method for signature 'deSet'
nullMatrix(object) <- value
```

Arguments

`object`	`S4 object`: `deSet`
`value`	`matrix`: null model matrix where columns are covariates and rows are observations

Value

`nullMatrix` returns the value of the null model matrix.

Author(s)

John Storey, Andrew Bass

See Also

`deSet`, `fullModel` and `fullModel`

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# extract the null model as a matrix
mat_null <- nullMatrix(de_obj)
```

`nullModel` *Null model equation from deSet object*

Description

These generic functions access and set the null model for `deSet` object.

Usage

```
nullModel(object)

nullModel(object) <- value

## S4 method for signature 'deSet'
nullModel(object)
```

```
## S4 replacement method for signature 'deSet'
nullModel(object) <- value
```

Arguments

object	S4 object: deSet
value	formula: The experiment design for the null model.

Value

nullModel returns the formula for the null model.

Author(s)

John Storey, Andrew Bass

See Also

deSet

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# extract the null model equation
mod_null <- nullModel(de_obj)

# change null model in experiment but must update full model
nullModel(de_obj) <- ~1
fullModel(de_obj) <- ~1 + ns(age, df=4)
```

odp *The optimal discovery procedure*

Description

odp performs the optimal discovery procedure, which is a framework for optimally performing many hypothesis tests in a high-dimensional study. When testing whether a feature is significant, the optimal discovery procedure uses information across all features when testing for significance.

Usage

```
odp(object, de.fit, odp.parms = NULL, weights = NULL, bs.its = 100,
  n.mods = 50, seed = NULL, verbose = TRUE, ...)

## S4 method for signature 'deSet,missing'
odp(object, de.fit, odp.parms = NULL,
  weights = NULL, bs.its = 100, n.mods = 50, seed = NULL,
  verbose = TRUE, ...)

## S4 method for signature 'deSet,deFit'
odp(object, de.fit, odp.parms = NULL,
  weights = NULL, bs.its = 100, n.mods = 50, seed = NULL,
  verbose = TRUE, ...)
```

Arguments

object	S4 object: deSet
de.fit	S4 object: deFit. Optional.
odp.parms	list: parameters for each cluster. See kl_clust.
weights	matrix: weights for each observation. Default is NULL.
bs.its	numeric: number of null bootstrap iterations. Default is 100.
n.mods	integer: number of clusters used in kl_clust. Default is 50.
seed	integer: set the seed value. Default is NULL.
verbose	boolean: print iterations for bootstrap method. Default is TRUE.
...	Additional arguments for qvalue and empPvals.

Details

The full ODP estimator computationally grows quadratically with respect to the number of genes. This becomes computationally taxing at a certain point. Therefore, an alternative method called mODP is used which has been shown to provide results that are very similar. mODP utilizes a clustering algorithm where genes are assigned to a cluster based on the Kullback-Leiber distance. Each gene is assigned an module-average parameter to calculate the ODP score and it reduces the computations time to approximately linear (see Woo, Leek and Storey 2010). If the number of clusters is equal to the number of genes then the original ODP is implemented. Depending on the number of hypothesis tests, this can take some time.

Value

deSet object

Author(s)

John Storey, Jeffrey Leek, Andrew Bass

References

Storey JD. (2007) The optimal discovery procedure: A new approach to simultaneous significance testing. Journal of the Royal Statistical Society, Series B, 69: 347-368.

Storey JD, Dai JY, and Leek JT. (2007) The optimal discovery procedure for large-scale significance testing, with applications to comparative microarray experiments. Biostatistics, 8: 414-432.

Woo S, Leek JT, Storey JD (2010) A computationally efficient modular optimal discovery procedure. Bioinformatics, 27(4): 509-515.

See Also

kl_clust, build_models and deSet

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov,
null.model = null_model, full.model = full_model)

# odp method
de_odp <- odp(de_obj, bs.its = 30)

# input a deFit object or ODP parameters ... not necessary
de_fit <- fit_models(de_obj, stat.type = "odp")
de_clust <- kl_clust(de_obj, n.mods = 10)
de_odp <- odp(de_obj, de.fit = de_fit, odp.parms = de_clust,
bs.its = 30)

# summarize object
summary(de_odp)
```

qvalueObj *Access/set qvalue slot*

Description

These generic functions access and set the qvalue object in the deSet object.

Usage

```
qvalueObj(object)

qvalueObj(object) <- value

## S4 method for signature 'deSet'
qvalueObj(object)

## S4 replacement method for signature 'deSet'
qvalueObj(object) <- value
```

Arguments

object	S4 object: deSet
value	S3 object: qvalue

Value

qvalueObj returns a qvalue object.

Author(s)

John Storey, Andrew Bass

See Also

lrt, odp and deSet

Examples

```
# import data
library(splines)
library(qvalue)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
```

```
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run the odp method
de_odp <- odp(de_obj, bs.its = 20)

# extract out significance results
qval_obj <- qvalueObj(de_odp)

# run qvalue and assign it to deSet slot
pvals <- qval_obj$pvalues
qval_new <- qvalue(pvals, pfdr = TRUE, fdr.level = 0.1)
qvalueObj(de_odp) <- qval_new
```

`resFull` *Residuals of full model fit*

Description

Access the fitted full model residuals in an `deFit` object.

Usage

```
resFull(object)

## S4 method for signature 'deFit'
resFull(object)
```

Arguments

`object`	S4 object: `deFit`

Value

`resFull` returns a matrix of residuals from full model.

Author(s)

John Storey, Andrew Bass

See Also

`fit_models`

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run fit_models to get model fits
de_fit <- fit_models(de_obj)

# extract out the full residuals from the model fit
res_full <- resFull(de_fit)
```

---

resNull *Residuals of null model fit*

Description

Access the fitted null model residuals in an deFit object.

Usage

```
resNull(object)

## S4 method for signature 'deFit'
resNull(object)
```

Arguments

object	S4 object: deFit

Value

resNull returns a matrix of residuals from null model.

Author(s)

John Storey, Andrew Bass

See Also

fit_models

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run fit_models to get model fits
de_fit <- fit_models(de_obj)

# extract out the null residuals from the model fits
res_null <- resNull(de_fit)
```

---

show *Show function for deFit and deSet*

Description

Show function for deFit and deSet objects.

Usage

```
show(object)

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

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

Arguments

object	S4 object: deSet

Value

Nothing of interest

Author(s)

John Storey, Andrew Bass

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# get summary
summary(de_obj)

# run odp and summarize
de_odp <- odp(de_obj, bs.its= 20)
de_odp
```

---

sType *Statistic type used in analysis*

Description

Access the statistic type in a deFit object. Can either be the optimal discovery procedure (odp) or the likelihood ratio test (lrt).

Usage

```
sType(object)

## S4 method for signature 'deFit'
sType(object)
```

Arguments

object	S4 object: deFit

Value

sType returns the statistic type- either "odp" or "lrt".

Author(s)

John Storey, Andrew Bass

See Also

fit_models, deFit and deSet

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# run fit_models to get model fits
de_fit <- fit_models(de_obj)

# extract the statistic type of model fits
stat_type <- sType(de_fit)
```

---

summary *Summary of deFit and deSet*

Description

Summary of deFit and deSet objects.

Usage

```
summary(object, ...)

## S4 method for signature 'deFit'
summary(object)
```

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

Arguments

object	S4 object: deSet
...	additional parameters

Value

Summary of deSet object

Author(s)

John Storey, Andrew Bass

Examples

```
# import data
library(splines)
data(kidney)
age <- kidney$age
sex <- kidney$sex
kidexpr <- kidney$kidexpr
cov <- data.frame(sex = sex, age = age)

# create models
null_model <- ~sex
full_model <- ~sex + ns(age, df = 4)

# create deSet object from data
de_obj <- build_models(data = kidexpr, cov = cov, null.model = null_model,
full.model = full_model)

# get summary
summary(de_obj)

# run odp and summarize
de_odp <- odp(de_obj, bs.its= 20)
summary(de_odp)
```

Index

∗ **datasets**
endotoxin, 13
gibson, 20
kidney, 23
∗ **nullModel,**
nullModel, 29
∗ **nullModel<-**
nullModel, 29
∗ **sType**
sType, 37
∗ **summary**
summary, 38

apply_qvalue, 3, *27*
apply_qvalue,deSet-method *(*apply_qvalue*)*, 3
apply_sva, 4
apply_sva,deSet-method *(*apply_sva*)*, 4

betaCoef, 5
betaCoef,deFit-method *(*betaCoef*)*, 5
build_models, 6, *8*, *27*, *32*
build_study, *7*, 7

deFit, *5*, *14, 15*, *17*, *24*, *26*, *31*, *34–38*
deFit-class, 9
deSet, *3–8*, *10*, 10, *16*, *18–20*, *22*, *24*, *26–33*, *36*, *38, 39*
deSet,ExpressionSet-method *(*deSet*)*, 10
deSet-class, 11

edge, 12
edge-package *(*edge*)*, 12
empPvals, *27*, *31*
endotoxin, 13
ExpressionSet, *10–12*

fit_models, *6*, *9*, *14*, *16*, 16, *25*, *27*, *34*, *36*, *38*
fit_models,deSet-method *(*fit_models*)*, 16
fitFull, 14
fitFull,deFit-method *(*fitFull*)*, 14
fitNull, 15
fitNull,deFit-method *(*fitNull*)*, 15
fullMatrix, 18
fullMatrix,deSet-method *(*fullMatrix*)*, 18
fullMatrix<- *(*fullMatrix*)*, 18
fullMatrix<-,deSet-method *(*fullMatrix*)*, 18
fullModel, *18*, 19, *29*
fullModel,deSet-method *(*fullModel*)*, 19
fullModel<- *(*fullModel*)*, 19
fullModel<-,deSet-method *(*fullModel*)*, 19

gibson, 20

individual, 22
individual,deSet-method *(*individual*)*, 22
individual<- *(*individual*)*, 22
individual<-,deSet-method *(*individual*)*, 22

kidney, 23
kl_clust, *17*, 24, *31, 32*
kl_clust,deSet,deFit-method *(*kl_clust*)*, 24
kl_clust,deSet,missing-method *(*kl_clust*)*, 24

lrt, *3*, *5*, *10*, *17*, 26, *33*
lrt,deSet,deFit-method *(*lrt*)*, 26
lrt,deSet,missing-method *(*lrt*)*, 26

nullMatrix, 28
nullMatrix,deSet-method *(*nullMatrix*)*, 28
nullMatrix<- *(*nullMatrix*)*, 28
nullMatrix<-,deSet-method *(*nullMatrix*)*, 28
nullModel, 29
nullModel,deSet-method *(*nullModel*)*, 29
nullModel<- *(*nullModel*)*, 29
nullModel<-,deSet-method *(*nullModel*)*, 29

odp, *3*, *5*, *10*, *17*, *25*, *27*, 31, *33*
odp,deSet,deFit-method *(*odp*)*, 31
odp,deSet,missing-method *(*odp*)*, 31

qvalue, *3*, *11, 12*, *27*, *31*, *33*
qvalueObj, 33
qvalueObj,deSet-method *(*qvalueObj*)*, 33
qvalueObj<- *(*qvalueObj*)*, 33
qvalueObj<-,deSet-method *(*qvalueObj*)*, 33

resFull, 34
resFull,deFit-method *(*resFull*)*, 34
resNull, 35
resNull,deFit-method *(*resNull*)*, 35

show, 36
show,deFit-method *(*show*)*, 36
show,deSet-method *(*show*)*, 36
sType, 37
sType,deFit-method *(*sType*)*, 37
summary, 38
summary,deFit-method *(*summary*)*, 38
summary,deSet-method *(*summary*)*, 38
sva, *4*, *12*