

# Package ‘edge’

October 12, 2016

**Type** Package

**Title** Extraction of Differential Gene Expression

**Date** 2015-04-15

**Version** 2.4.2

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

**Maintainer** John D. Storey <jstorey@princeton.edu>, Andrew J. Bass  
<ajbass@princeton.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 snm, sva, and qvalue are integrated in edge to provide a wide range of tools for gene expression analysis.

**VignetteBuilder** knitr

**Imports** methods, splines, sva, snm, jackstraw, 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

**R topics documented:**

|                           |           |
|---------------------------|-----------|
| apply_jackstraw . . . . . | 2         |
| apply_qvalue . . . . .    | 4         |
| apply_snm . . . . .       | 6         |
| apply_sva . . . . .       | 7         |
| betaCoef . . . . .        | 8         |
| build_models . . . . .    | 9         |
| build_study . . . . .     | 10        |
| deFit-class . . . . .     | 12        |
| deSet . . . . .           | 13        |
| deSet-class . . . . .     | 14        |
| edge . . . . .            | 15        |
| endotoxin . . . . .       | 16        |
| fitFull . . . . .         | 17        |
| fitNull . . . . .         | 18        |
| fit_models . . . . .      | 19        |
| fullMatrix . . . . .      | 21        |
| fullModel . . . . .       | 22        |
| gibson . . . . .          | 23        |
| individual . . . . .      | 25        |
| kidney . . . . .          | 26        |
| kl_clust . . . . .        | 27        |
| lrt . . . . .             | 29        |
| nullMatrix . . . . .      | 31        |
| nullModel . . . . .       | 32        |
| odp . . . . .             | 34        |
| qvalueObj . . . . .       | 36        |
| resFull . . . . .         | 37        |
| resNull . . . . .         | 38        |
| show . . . . .            | 39        |
| sType . . . . .           | 40        |
| summary . . . . .         | 41        |
| <b>Index</b>              | <b>43</b> |

---

|                 |  |
|-----------------|--|
| apply_jackstraw | <i>Non-Parametric Jackstraw for Principal Component Analysis (PCA)</i> |
|-----------------|--|

---

**Description**

Estimates statistical significance of association between variables and their principal components (PCs).

**Usage**

```
apply_jackstraw(object, r1 = NULL, r = NULL, s = NULL, B = NULL,
  covariate = NULL, verbose = TRUE, seed = NULL)
```

```
## S4 method for signature 'deSet'
```

```
apply_jackstraw(object, r1 = NULL, r = NULL, s = NULL,
  B = NULL, covariate = NULL, verbose = TRUE, seed = NULL)
```

**Arguments**

|           |  |
|-----------|--|
| object    | S4 object: <a href="#">deSet</a>   |
| r1        | a numeric vector of principal components of interest. Choose a subset of r significant PCs to be used.                 |
| r         | a number (a positive integer) of significant principal components.   |
| s         | a number (a positive integer) of synthetic null variables. Out of m variables, s variables are independently permuted. |
| B         | a number (a positive integer) of resampling iterations. There will be a total of s*B null statistics.                  |
| covariate | a data matrix of covariates with corresponding n observations.   |
| verbose   | a logical indicator as to whether to print the progress.   |
| seed      | a seed for the random number generator.  |

**Details**

This function computes m p-values of linear association between m variables and their PCs. Its resampling strategy accounts for the over-fitting characteristics due to direct computation of PCs from the observed data and protects against an anti-conservative bias.

Provide the [deSet](#), with m variables as rows and n observations as columns. Given that there are r significant PCs, this function tests for linear association between m variables and their r PCs.

You could specify a subset of significant PCs that you are interested in r1. If PC is given, then this function computes statistical significance of association between m variables and PC, while adjusting for other PCs (i.e., significant PCs that are not your interest). For example, if you want to identify variables associated with 1st and 2nd PCs, when your data contains three significant PCs, set r=3 and r1=c(1,2).

Please take a careful look at your data and use appropriate graphical and statistical criteria to determine a number of significant PCs, r. The number of significant PCs depends on the data structure and the context. In a case when you fail to specify r, it will be estimated from a permutation test (Buja and Eyuboglu, 1992) using a function [permutationPA](#).

If s is not supplied, s is set to about 10 supplied, B is set to m\*10/s.

**Value**

apply\_jackstraw returns a list containing the following slots:

- p.value the m p-values of association tests between variables and their principal components
- obs.stat the observed F-test statistics
- null.stat the s\*B null F-test statistics

**Author(s)**

Neo Christopher Chung <nc@princeton.edu>

**References**

Chung and Storey (2013) Statistical Significance of Variables Driving Systematic Variation in High-Dimensional Data. arXiv:1308.6013 [stat.ME] <http://arxiv.org/abs/1308.6013>

More information available at <http://ncc.name/>

**See Also**

[permutationPA](#)

**Examples**

```
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)
## apply the jackstraw
out = apply_jackstraw(de_obj, r1=1, r=1)
## Use optional arguments
## For example, set s and B for a balance between speed of the algorithm and accuracy of p-values
## out = apply_jackstraw(dat, r1=1, r=1, s=10, B=1000, seed=5678)
```

---

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_snm`*Supervised normalization of data in edge*

---

### Description

Runs `snm` on a `deSet` object based on the null and full models in `deSet`. See `snm` for additional details on the algorithm.

### Usage

```
apply_snm(object, int.var = NULL, ...)  
  
## S4 method for signature 'deSet'  
apply_snm(object, int.var = NULL, ...)
```

### Arguments

|                      |  |
|----------------------|--|
| <code>object</code>  | S4 object: <code>deSet</code>  |
| <code>int.var</code> | data frame: intensity-dependent effects (see <code>snm</code> for details) |
| <code>...</code>     | Additional arguments for <code>snm</code>                                  |

### Value

`apply_snm` returns a `deSet` object where `assayData` (the expression data) that has been passed to `apply_snm` is replaced with the normalized data that `snm` returns. Specifically, `exprs(object)` is replaced by `$norm.dat` from `snm`, where `object` is the `deSet` object.

### Author(s)

John Storey, Andrew Bass

### References

Mechan BH, Nelson PS, Storey JD. Supervised normalization of microarrays. *Bioinformatics* 2010;26:1308-1315.

### See Also

`deSet`, `odp` and `lrt`

### Examples

```
# simulate data  
library(snm)  
singleChannel <- sim.singleChannel(12345)  
data <- singleChannel$raw.data  
  
# create deSet object using build_models (can use ExpressionSet see manual)
```

```
cov <- data.frame(grp = singleChannel$bio.var[,2])
full_model <- ~grp
null_model <- ~1

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

# run snm using intensity-dependent adjustment variable
de_snm <- apply_snm(de_obj, int.var = singleChannel$int.var,
verbose = FALSE, num.iter = 1)
```

---

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: <a href="#">deSet</a>             |
| ...    | Additional arguments for <a href="#">sva</a> |

## 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 assignment 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: <a href="#">ExpressionSet</a>  |
| 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_snm(deSet, ...)` Applies supervised normalization of microarrays ([snm](#)) on gene expression data.

`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 experiment
# 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: <a href="#">deSet</a> .  |
| 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 | <i>Matrix representation of full model</i> |
|------------|--|

---

### 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: <a href="#">deSet</a>   |
| 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 (VIL-LAGE), 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 | <i>Individuals sampled in experiment</i> |
|------------|--|

---

### 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 | <a href="#">deSet</a>   |
| 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 distinct 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 experiment
# 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: <a href="#">deSet</a> .           |
| de.fit | S4 object: <a href="#">deFit</a> .           |
| 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  $n.mods \geq 50$  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: <a href="#">deSet</a> .           |
| de.fit | S4 object: <a href="#">deFit</a> . 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 <a href="#">empPvals</a> ). 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 <a href="#">apply_qvalue</a> and <a href="#">empPvals</a> 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](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: <a href="#">deSet</a>   |
| 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: <a href="#">deSet</a>                   |
| 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)
```

**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: <a href="#">deSet</a>   |
| de.fit    | S4 object: <a href="#">deFit</a> . Optional.                                   |
| odp.parms | list: parameters for each cluster. See <a href="#">kl_clust</a> .              |
| 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 <a href="#">kl_clust</a> . 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 <a href="#">qvalue</a> and <a href="#">empPvals</a> . |

**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 | <i>Access/set qvalue slot</i> |
|-----------|-------------------------------|

---

### 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: <a href="#">deSet</a>  |
| value  | S3 object: <a href="#">qvalue</a> |

### 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
...         additional parameters
```

**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)
```

**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

- \*Topic **datasets**
  - endotoxin, 16
  - gibson, 23
  - kidney, 26
- \*Topic **nullModel**,
  - nullModel, 32
- \*Topic **nullModel<-**
  - nullModel, 32
- \*Topic **sType**
  - sType, 40
- \*Topic **summary**
  - summary, 41
- apply\_jackstraw, 2
- apply\_jackstraw, deSet-method (apply\_jackstraw), 2
- apply\_qvalue, 4, 30
- apply\_qvalue, deSet-method (apply\_qvalue), 4
- apply\_snm, 6
- apply\_snm, deSet-method (apply\_snm), 6
- apply\_sva, 7
- apply\_sva, deSet-method (apply\_sva), 7
- betaCoef, 8
- betaCoef, deFit-method (betaCoef), 8
- build\_models, 9, 11, 30, 35
- build\_study, 10, 10
- deFit, 8, 17, 18, 20, 27, 29, 34, 37–41
- deFit-class, 12
- deSet, 3–11, 13, 13, 19, 21–23, 25, 27, 29–36, 39, 41, 42
- deSet, ExpressionSet-method (deSet), 13
- deSet-class, 14
- edge, 15
- edge-package (edge), 15
- empPvals, 30, 34
- endotoxin, 16
- ExpressionSet, 13–15
- fit\_models, 9, 12, 17, 19, 19, 28, 30, 37, 39, 41
- fit\_models, deSet-method (fit\_models), 19
- fitFull, 17
- fitFull, deFit-method (fitFull), 17
- fitNull, 18
- fitNull, deFit-method (fitNull), 18
- fullMatrix, 21
- fullMatrix, deSet-method (fullMatrix), 21
- fullMatrix<- (fullMatrix), 21
- fullMatrix<- , deSet-method (fullMatrix), 21
- fullModel, 21, 22, 32
- fullModel, deSet-method (fullModel), 22
- fullModel<- (fullModel), 22
- fullModel<- , deSet-method (fullModel), 22
- gibson, 23
- individual, 25
- individual, deSet-method (individual), 25
- individual<- (individual), 25
- individual<- , deSet-method (individual), 25
- kidney, 26
- kl\_clust, 20, 27, 34, 35
- kl\_clust, deSet, deFit-method (kl\_clust), 27
- kl\_clust, deSet, missing-method (kl\_clust), 27
- lrt, 5, 6, 8, 13, 20, 29, 36
- lrt, deSet, deFit-method (lrt), 29
- lrt, deSet, missing-method (lrt), 29
- nullMatrix, 31
- nullMatrix, deSet-method (nullMatrix), 31
- nullMatrix<- (nullMatrix), 31

`nullMatrix<-`, `deSet`-method (`nullMatrix`),  
31  
`nullModel`, 32  
`nullModel`, `deSet`-method (`nullModel`), 32  
`nullModel<-` (`nullModel`), 32  
`nullModel<-`, `deSet`-method (`nullModel`), 32  
  
`odp`, 5, 6, 8, 13, 20, 28, 30, 34, 36  
`odp`, `deSet`, `deFit`-method (`odp`), 34  
`odp`, `deSet`, `missing`-method (`odp`), 34  
  
`permutationPA`, 3, 4  
  
`qvalue`, 4, 5, 14, 15, 30, 34, 36  
`qvalueObj`, 36  
`qvalueObj`, `deSet`-method (`qvalueObj`), 36  
`qvalueObj<-` (`qvalueObj`), 36  
`qvalueObj<-`, `deSet`-method (`qvalueObj`), 36  
  
`resFull`, 37  
`resFull`, `deFit`-method (`resFull`), 37  
`resNull`, 38  
`resNull`, `deFit`-method (`resNull`), 38  
  
`show`, 39  
`show`, `deFit`-method (`show`), 39  
`show`, `deSet`-method (`show`), 39  
`snm`, 6, 15  
`sType`, 40  
`sType`, `deFit`-method (`sType`), 40  
`summary`, 41  
`summary`, `deFit`-method (`summary`), 41  
`summary`, `deSet`-method (`summary`), 41  
`sva`, 7, 15