# Package 'OUTRIDER'

November 17, 2026

**Title** OUTRIDER - OUTlier in RNA-Seq fInDER

**Type** Package

**Version** 1.31.0

**Date** 2026-02-10

**URL** https://github.com/gagneurlab/OUTRIDER

**BugReports** https://github.com/gagneurlab/OUTRIDER/issues

**Description** Identification of aberrant gene expression in RNA-seq data. Read count expectations are modeled by an autoencoder to control for confounders in the data. Given these expectations, the RNA-seq read counts are assumed to follow a negative binomial distribution with a gene-specific dispersion. Outliers are then identified as read counts that significantly deviate from this distribution. Furthermore, OUTRIDER provides useful plotting functions to analyze and visualize the results.

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, RNASeq, Transcriptomics, Alignment, Sequencing, GeneExpression, Genetics

**License** file LICENSE

**NeedsCompilation** yes

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 3.6), BiocParallel, GenomicFeatures, SummarizedExperiment, methods

**Imports** BBmisc, BiocGenerics, data.table, DESeq2 (>= 1.16.1), generics, GenomicRanges, ggplot2, ggrepel, graphics, grDevices, heatmaply, IRanges, matrixStats, pcaMethods, pheatmap, plotly, plyr, pracma, PRROC, RColorBrewer, reshape2, RMTstat, S4Vectors, scales, splines, stats, txdbmaker, utils

**Suggests** testthat, knitr, rmarkdown, BiocStyle, TxDb.Hsapiens.UCSC.hg19.knownGene, org.Hs.eg.db, RMariaDB, AnnotationDbi, beeswarm, covr, GenomeInfoDb, ggbio, biovizBase

**LinkingTo** Rcpp, RcppArmadillo

**Collate** package-OUTRIDER.R class-OutriderDataSet.R AllGenerics.R inputCheckerFunctions.R helperFunctions.R getNSetterFuns.R getNSetterFunsInternal.R autoencoder.R fitNB.R ZscoreMatrix.R

method-evaluation.R method-counts.R method-estimateBestQ.R
method-plot.R method-results.R pValMatrix.R filterExpression.R
OUTRIDER.R sizeFactor.R RcppExports.R updateE.R updateD.R
updateTheta.R PCAcorrections.R thetaMethodOfMoments.R
controlForConfounders.R lossNGradientRVersion.R

**git_url** https://git.bioconductor.org/packages/OUTRIDER

**git_branch** devel

**git_last_commit** f5f7a6b

**git_last_commit_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-11-16

**Author** Felix Brechtmann [aut] (ORCID: <https://orcid.org/0000-0002-0110-152X>),
Christian Mertes [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1091-205X>),
Agne Matuseviciute [aut],
Michaela Fee Müller [ctb],
Andrea Raithel [ctb],
Vicente Yepez [aut] (ORCID: <https://orcid.org/0000-0001-7916-3643>),
Julien Gagneur [aut] (ORCID: <https://orcid.org/0000-0002-8924-8365>)

**Maintainer** Christian Mertes <mertes@in.tum.de>

# Contents

aberrant . . . 3
computeGeneLength . . . 4
computeLatentSpace . . . 4
computePvalues . . . 5
computeZscores . . . 6
controlForConfounders . . . 7
counts . . . 8
estimateBestQ . . . 9
filterExpression . . . 10
fit . . . 12
fpkm . . . 12
getBestQ . . . 13
makeExampleOutriderDataSet . . . 14
normalizationFactors . . . 15
OUTRIDER . . . 16
OutriderDataSet-class . . . 17
plotAberrantPerSample . . . 18
results . . . 26
sampleExclusionMask . . . 28
sizeFactors . . . 28

**Index** **30**

aberrant *Number of aberrant events*

## Description

Identifies the aberrant events and returns the number of aberrant counts per gene or sample or returns a matrix indicating aberrant events.

## Usage

```
aberrant(object, ...)

## S4 method for signature 'OutriderDataSet'
aberrant(
  object,
  padjCutoff = 0.05,
  zScoreCutoff = 0,
  by = c("none", "sample", "gene"),
  subsetName = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| object | An OutriderDataSet object |
| ... | Currently not in use. |
| padjCutoff | The padjust cutoff |
| zScoreCutoff | The absolute Z-score cutoff, if NA or NULL no Z-score cutoff is used |
| by | if the results should be summarized by 'sample', 'gene' or not at all (default). |
| subsetName | The name of a subset of genes of interest for which FDR corected pvalues were previously computed. Those FDR values on the subset will then be used to determine aberrant status. Default is NULL (using transcriptome-wide FDR corrected pvalues). |

## Value

The number of aberrent events by gene or sample or a TRUE/FALSE matrix of the size sample x gene of aberrent events.

## Examples

```
ods <- makeExampleOutriderDataSet()
ods <- OUTRIDER(ods, implementation='pca')

aberrant(ods)[1:10,1:10]
tail(sort(aberrant(ods, by="sample")))
tail(sort(aberrant(ods, by="gene")))
```

## computeGeneLength *Extracting the gene length from annotations*

### Description

Computes the length for each gene based on the given GTF file or annotation. Here the length of a gene is defind by the total number of bases covered by exons.

### Usage

```
computeGeneLength(ods, gtfFile, format = "gtf", mapping = NULL, ...)
```

### Arguments

| | |
|---|---|
| ods | An OutriderDataSet for which the gene length should be computed. |
| gtfFile | Can be a GTF file or an txDb object with annotation. |
| format | The format parameter from makeTxDbFromGFF |
| mapping | If set, it is used to map gene names between the GFT and the ods object. This should be a 2 column data.frame: 1. column GTF names and 2. column ods names. |
| ... | further arguments to makeTxDbFromGFF |

### Value

An OutriderDataSet containing a basepairs column with the calculated gene length. Accessable through mcols(ods)['baisepairs']

### Examples

```
ods <- makeExampleOutriderDataSet(dataset="GTExSkinSmall")
annotationFile <- system.file("extdata", "gencode.v19.genes.small.gtf.gz",
        package="OUTRIDER")
ods <- computeGeneLength(ods, annotationFile)

mcols(ods)['basepairs']
fpkm(ods)[1:10,1:10]
```

## computeLatentSpace *Extracting the latent space*

### Description

Extracts the latent space from the OutriderDataSet object determined by the autoencoder.

### Usage

```
computeLatentSpace(ods)
```

### Arguments

| | |
|---|---|
| ods | An OutriderDataSet |

### Value

A matrix containing the latent space determined by the autoencoder.

### Examples

```
ods <- makeExampleOutriderDataSet()

ods <- estimateSizeFactors(ods)
ods <- controlForConfounders(ods, implementation="pca")
computeLatentSpace(ods)[,1:6]
```

---

## computePvalues *Calculate P-values*

---

### Description

This function computes the P-values based on the fitted negative binomial model being an outlier for the given sample gene combination. It computes two matrices with the same dimension as the count matrix (samples x genes), which contain the corresponding P-values and adjusted P-values of every count. The adjusted P-values are computed across all genes per sample.

### Usage

```
computePvalues(object, ...)

## S4 method for signature 'OutriderDataSet'
computePvalues(
  object,
  alternative = c("two.sided", "greater", "less"),
  method = "BY",
  subsets = NULL,
  BPPARAM = bpparam()
)
```

### Arguments

| | |
|---|---|
| object | An OutriderDataSet |
| ... | additional params, currently not used. |
| alternative | Can be one of "two.sided", "greater" or "less" to specify the alternative hypothesis used to calculate the P-values, defaults to "two.sided" |
| method | Method used for multiple testing |
| subsets | A named list of named lists specifying any number of gene subsets (can differ per sample). For each subset, FDR correction will be limited to genes in the subset, and the FDR corrected pvalues stored as an assay in the ods object in addition to the transcriptome-wide FDR corrected pvalues. See the examples for how to use this argument. |
| BPPARAM | Can be used to parallelize the computation, defaults to bpparam() |

## Value

An OutriderDataSet object with computed nominal and adjusted P-values

## See Also

p.adjust

## Examples

```
ods <- makeExampleOutriderDataSet()

ods <- estimateSizeFactors(ods)
ods <- fit(ods)

ods <- computePvalues(ods)

assays(ods)[['pValue']][1:10,1:10]

# example of restricting FDR correction to subsets of genes of interest
genesOfInterest <- list("sample_1"=sample(rownames(ods), 3),
                        "sample_2"=sample(rownames(ods), 8),
                        "sample_6"=sample(rownames(ods), 5))
ods <- computePvalues(ods, subsets=list("exampleSubset"=genesOfInterest))
padj(ods, subsetName="exampleSubset")[1:10,1:10]
ods <- computePvalues(ods,
                subsets=list("anotherExampleSubset"=rownames(ods)[1:5]))
padj(ods, subsetName="anotherExampleSubset")[1:10,1:10]
```

---

# computeZscores *Z score computation*

---

## Description

Computes the z scores for every count in the matrix. The z score is defined in the $\log_2$ space as follows: $z_{ij} = \frac{l_{ij} - \mu_j^l}{\sigma_j^l}$ where `l` is the $\log_2$ transformed normalized count and $\mu$ and $\sigma$ the mean and standard deviation for gene `j` and sample `i`, respectively.

## Usage

```
computeZscores(ods, ...)

## S4 method for signature 'OutriderDataSet'
computeZscores(ods, peerResiduals = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `ods` | OutriderDataSet |
| `...` | Further arguments passed on to `ZscoreMatrix`. |
| `peerResiduals` | If TRUE, PEER residuals are used to compute Z scores |

## Value

An OutriderDataSet containing the Z score matrix "zScore" and the log2 fold changes "l2fc" as asasys.

## Examples

```
ods <- makeExampleOutriderDataSet()
ods <- estimateSizeFactors(ods)

ods <- controlForConfounders(ods, implementation="pca")
ods <- computeZscores(ods)

zScore(ods)[1:10,1:10]
assay(ods, "l2fc")[1:10,1:10]
```

---

# controlForConfounders *Autoencoder function to correct for confounders.*

---

## Description

This is the wrapper function for the autoencoder implementation. It can be used to call the standard R implementation or the experimental Python implementation.

## Usage

```
controlForConfounders(
  ods,
  q,
  implementation = c("autoencoder", "pca"),
  BPPARAM = bpparam(),
  ...
)
```

## Arguments

| | |
|---|---|
| ods | An OutriderDataSet object |
| q | The encoding dimensions |
| implementation | "autoencoder", the default, will use the autoencoder implementation. Also 'pca' and 'peer' can be used to control for confounding effects |
| BPPARAM | A BiocParallelParam instance to be used for parallel computing. |
| ... | Further arguments passed on to the specific implementation method. |

## Value

An ods object including the control factors

## Examples

```
ods <- makeExampleOutriderDataSet()
implementation <- 'autoencoder'

ods <- estimateSizeFactors(ods)
ods <- controlForConfounders(ods, implementation=implementation)

plotCountCorHeatmap(ods, normalized=FALSE)
plotCountCorHeatmap(ods, normalized=TRUE)
```

---

`counts` *Accessors for the 'counts' slot of an OutriderDataSet object.*

---

## Description

The counts slot holds the count data as a matrix of non-negative integer count values, one row for each observational unit (eg.: gene), and one column for each sample.

## Usage

```
## S4 method for signature 'OutriderDataSet'
counts(object, normalized = FALSE, minE = 0.5, ...)

## S4 replacement method for signature 'OutriderDataSet,matrix'
counts(object, ...) <- value
```

## Arguments

| | |
|---|---|
| `object` | An `OutriderDataSet` object |
| `normalized` | TRUE/FALSE whether counts should be normalized |
| `minE` | The minimal expected count, defaults to 0.5, to be used in computing the expected log geom mean. |
| `...` | Further arguments are passed on to the underlying assay function |
| `value` | An integer matrix containing the counts |

## Details

By default this function returns the raw counts. If conrol factors are computed or provided the normalized counts can be returned using normalized = TRUE. The offset parameter can be used to add a pseudocount to the count before dividing by the normalization. This can be usefull when the log(counts) are computed and in case the controll values are in the same oder of magnited as the counts.

## Value

A matrix containing the counts

## See Also

`sizeFactors`, `normalizationFactors`

## Examples

```
ods <- makeExampleOutriderDataSet()
counts(ods)[1:10,1:10]

ods <- estimateSizeFactors(ods)
counts(ods, normalized=TRUE)[1:10,1:10]
```

---

`estimateBestQ` *Find the optimal encoding dimension*

---

## Description

Finds the optimal encoding dimension by either Optimal Hard Thresholding or injecting artificial splicing outlier ratios while maximizing the precision-recall curve.

## Usage

```
estimateBestQ(
  ods = NULL,
  zScoresOHT = NULL,
  useOHT = TRUE,
  params = seq(2, min(100, ncol(ods) - 1, nrow(ods) - 1), 2),
  freq = 0.01,
  zScore = 3,
  sdlog = log(1.6),
  lnorm = TRUE,
  inj = "both",
  ...,
  BPPARAM = bpparam()
)

findInjectZscore(
  ods,
  freq = 0.01,
  zScoreParams = c(seq(1.5, 4, 0.5), "lnorm"),
  encDimParams = c(seq(3, 40, 3), seq(45, 70, 5), 100, 130, 160),
  inj = "both",
  ...,
  BPPARAM = bpparam()
)
```

## Arguments

| | |
|---|---|
| `ods` | An OutriderDataSet object |
| `zScoresOHT` | A z-score matrix |
| `useOHT` | If `TRUE` (default), Optimal Hard Thresholding is used to estimate the optimal encoding dimension. |
| `params`, `encDimParams` | Set of possible q values. |

| | |
|---|---|
| freq | Frequency of outlier, defaults to 1E-2 |
| zScore, zScoreParams | Set of possible injection Z-score, defaults to 3. |
| sdlog | Standard deviation of the sitribution on the log scale. |
| lnorm | If TRUE, the default, Z-scores are drawn from a log normal distribution with a mean of log(zScore) in log-scale. |
| inj | Injection strategy, by default 'both'. |
| ... | Further arguments passed on to the controlForConfounders function. |
| BPPARAM | BPPARAM object by default bpparam(). |

## Value

The OutriderDataSet object with the optimal encoding dimension saved in the metadata

## Examples

```
ods <- makeExampleOutriderDataSet()

# run OHT (default)
estimateBestQ(ods)

# run hyperparameter optimization (grid-search)
encDimSearchParams <- c(5, 8, 10, 12, 15)
zScoreParams <- c(2, 3, 5, 'lnorm')
implementation <- 'autoencoder'
register(MulticoreParam(4))

ods1 <- estimateBestQ(ods, useOHT=FALSE, params=encDimSearchParams,
        implementation=implementation)
plotEncDimSearch(ods1)

ods2 <- findInjectZscore(ods, zScoreParams=zScoreParams,
        encDimParams=encDimSearchParams, implementation=implementation)
plotEncDimSearch(ods2)
```

---

# filterExpression *Filter expression*

---

## Description

To filter out non expressed genes this method uses the FPKM values to get a comparable value over genes. For each gene, if the pth- percentile is greater than the fpkmCutoff value, it passes the filter. To calcute the FPKM values the user needs to provide a GTF file or the basepair parameter as described in fpkm.

## Usage

```
filterExpression(object, ...)

## S4 method for signature 'OutriderDataSet'
filterExpression(
  object,
  gtfFile,
  fpkmCutoff = 1,
  percentile = 0.95,
  filterGenes = TRUE,
  savefpkm = FALSE,
  minCounts = FALSE,
  addExpressedGenes = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| object | An OutriderDataSet object |
| ... | Additional arguments passed to computeGeneLength |
| gtfFile | A txDb object or a GTF/GFF file to be used as annotation |
| fpkmCutoff | The threshold for filtering based on the FPKM value |
| percentile | a numeric indicating the percentile FPKM value to compare against the fpkmCutoff |
| filterGenes | If TRUE, the default, the object is subseted. |
| savefpkm | If TRUE, the FPKM values are saved as assay |
| minCounts | If TRUE, only genes with 0 counts in all samples are filtered |
| addExpressedGenes | If TRUE (default), adds 5 columns to the colData with information regarding the number of expressed genes per sample |

## Value

An OutriderDataSet containing the passedFilter column, which indicates if the given gene passed the filtering threshold. If filterGenes is TRUE the object is already subsetted.

## Examples

```
ods <- makeExampleOutriderDataSet(dataset="GTExSkinSmall")
annotationFile <- system.file("extdata",
    "gencode.v19.genes.small.gtf.gz", package="OUTRIDER")
ods <- filterExpression(ods, annotationFile, filterGenes=FALSE)

mcols(ods)['passedFilter']
fpkm(ods)[1:10,1:10]
dim(ods)

ods <- ods[mcols(ods)[['passedFilter']]]
dim(ods)
```

## fit *Fit the negative binomial distribution*

### Description

Fit a negative binomial (NB) distribution to the counts per gene over all samples using, if available, the precomputed control factors. If no normalization factors are provided only the sizeFactors are used.

### Usage

```
## S3 method for class 'OutriderDataSet'
fit(object, BPPARAM = bpparam(), ...)
```

### Arguments

| | |
|---|---|
| object | An OutriderDataSet |
| BPPARAM | by default bpparam() |
| ... | Currently not used. |

### Value

An OutriderDataSet object with the fitted model. Accessible through: `mcols(ods)[,c('mu', 'theta')]`.

### Examples

```
ods <- makeExampleOutriderDataSet()
ods <- estimateSizeFactors(ods)
ods <- fit(ods)

mcols(ods)[1:10,c('mu', 'theta')]
```

## fpkm *Calculate FPM and FPKM values*

### Description

This is the fpm and fpkm function from DESeq2. For more details see: `fpkm` and `fpm`

### See Also

`fpkm` `fpm`

## Examples

```
ods <- makeExampleOutriderDataSet()
mcols(ods)['basepairs'] <- round(rnorm(nrow(ods), 1000, 500))

mcols(ods)['basepairs']
fpkm(ods)[1:10,1:10]
fpm(ods)[1:10,1:10]
```

---

`getBestQ` *Getter/Setter functions*

---

## Description

This is a collection of small accessor/setter functions for easy access to the values within the OUTRIDER model.

## Usage

```
getBestQ(ods)

zScore(ods)

pValue(ods)

padj(ods, subsetName = NULL)

## S4 method for signature 'OutriderDataSet'
dispersions(object, ...)

theta(ods)
```

## Arguments

| | |
|---|---|
| `ods, object` | An OutriderDataSet object. |
| `subsetName` | Name of a gene subset for which to store or retrieve FDR corrected p values |
| `...` | Further arguments passed on to the underlying assay function. |

## Value

A matrix or vector dependent on the type of data retrieved.

## See Also

`estimateDispersions`

## Examples

```
ods <- makeExampleOutriderDataSet(10, 10)
ods <- OUTRIDER(ods)

zScore(ods)
pValue(ods)
padj(ods)
theta(ods)
theta(ods) == 1/dispersions(ods)
getBestQ(ods)
```

---

`makeExampleOutriderDataSet` *Create example data sets for OUTRIDER*

---

## Description

Creates an example data set from a file or simulates a data set based on random counts following a negative binomial distribution with injected outliers with a fixed z score away from the mean of the gene.

## Usage

```
makeExampleOutriderDataSet(
  n = 200,
  m = 80,
  q = 10,
  freq = 0.001,
  zScore = 6,
  inj = c("both", "low", "high"),
  sf = rnorm(m, mean = 1, sd = 0.1),
  dataset = c("none", "GTExSkinSmall", "KremerNBaderSmall")
)
```

## Arguments

| | |
|---|---|
| `n` | Number of simulated genes |
| `m` | Number of simulated samples |
| `q` | number of simulated latend variables. |
| `freq` | Frequency of in-silico outliers |
| `zScore` | Absolute z score of in-silico outliers (default 6). |
| `inj` | Determines whether counts are injected with the strategy ('both', 'low', 'high'), default is 'both'. |
| `sf` | Artificial Size Factors |
| `dataset` | If "none", the default, an example data set is simulated. One can also use example data set included in the package by specifying 'GTExSkinSmall' or 'KremerNBaderSmall' |

## Value

An OutriderDataSet containing an example dataset. Depending on the parameters it is based on a real data set or it is simulated

## Examples

```
# A generic dataset
ods1 <- makeExampleOutriderDataSet()
ods1

# A generic dataset with specificed sample size and injection method
ods2 <- makeExampleOutriderDataSet(n=200, m=50, inj='low')
ods2

# A subset of a real world dataset from GTEx
ods3 <- makeExampleOutriderDataSet(dataset="GTExSkinSmall")
ods3
```

---

# normalizationFactors *Accessor functions for the normalization factors in an OutriderDataSet object.*

---

## Description

To normalize raw count data normalization factors can be provided as a matrix. When running controlForConfounders the normalization factors are stored within the OutriderDataset object. This normalization factors are then used to compute the normalized counts.

## Usage

```
## S4 method for signature 'OutriderDataSet'
normalizationFactors(object, ...)

## S4 replacement method for signature 'OutriderDataSet,matrix'
normalizationFactors(object, minE = 0.5, ...) <- value

## S4 replacement method for signature 'OutriderDataSet,DataFrame'
normalizationFactors(object, minE = 0.5, ...) <- value

## S4 replacement method for signature 'OutriderDataSet,data.frame'
normalizationFactors(object, minE = 0.5, ...) <- value

## S4 replacement method for signature 'OutriderDataSet,NULL'
normalizationFactors(object) <- value
```

## Arguments

| | |
|---|---|
| object | An OutriderDataSet object |
| ... | Further arguments are passed on to the underlying assay function |
| minE | The minimal expected count, defaults to 0.5, to be used in computing the expected log geom mean. |
| value | The matrix of normalization factors |

## Value

A numeric matrix containing the normalization factors or the OutriderDataSet object with an updated `normalizationFactors` assay.

## See Also

`sizeFactors` `normalizationFactors`

## Examples

```
ods <- makeExampleOutriderDataSet()

normFactors <- matrix(runif(nrow(ods)*ncol(ods),0.5,1.5),
    ncol=ncol(ods),nrow=nrow(ods))

# the normalization factors matrix should not have 0's in it
# it should have geometric mean near 1 for each row
normFactorsRM <- normFactors / exp(rowMeans(log(normFactors)))
normalizationFactors(ods) <- normFactorsRM
normalizationFactors(ods)[1:10,1:10]

normalizationFactors(ods) <- NULL
ods <- estimateSizeFactors(ods)
normalizationFactors(ods) <- normFactors
all(normalizationFactors(ods) == t(sizeFactors(ods) * t(normFactors)))
```

---

`OUTRIDER` *OUTRIDER - Finding expression outlier events*

---

## Description

The OUTRIDER function runs the default OUTRIDER pipeline combinig the fit, the computation of Z scores and P-values. All computed values are returned as an OutriderDataSet object.

To have more control over each analysis step, one can call each function separately.

1. `estimateSizeFactors` to calculate the sizeFactors
2. `controlForConfounders` to control for confounding effects
3. `fit` to fit the negative binomial model (only needed if the autoencoder is not used)
4. `computePvalues` to calculate the nominal and adjusted P-values
5. `computeZscores` to calculate the Z scores

## Usage

```
OUTRIDER(
  ods,
  q,
  controlData = TRUE,
  implementation = "autoencoder",
  subsets = NULL,
  BPPARAM = bpparam(),
  ...
)
```

### Arguments

| | |
|---|---|
| ods | An OutriderDataSet object |
| q | The encoding dimensions |
| controlData | If TRUE, the default, the raw counts are controled for confounders by the autoencoder |
| implementation | "autoencoder", the default, will use the autoencoder implementation. Also 'pca' and 'peer' can be used to control for confounding effects |
| subsets | A named list of named lists specifying any number of gene subsets (can differ per sample). For each subset, FDR correction will be limited to genes in the subset, and the FDR corrected pvalues stored as an assay in the ods object in addition to the transcriptome-wide FDR corrected pvalues. See the examples for how to use this argument. |
| BPPARAM | A BiocParallelParam instance to be used for parallel computing. |
| ... | Further arguments passed on to controlForConfounders |

### Value

OutriderDataSet with all the computed values. The values are stored as assays and can be accessed by: `assay(ods, 'value')`. To get a full list of calculated values run: `assayNames(ods)`

### Examples

```
ods <- makeExampleOutriderDataSet()
implementation <- 'autoencoder'

ods <- OUTRIDER(ods, implementation=implementation)

pValue(ods)[1:10,1:10]
res <- results(ods, all=TRUE)
res

plotAberrantPerSample(ods)
plotVolcano(ods, 1)

# example of restricting FDR correction to subsets of genes of interest
genesOfInterest <- list("sample_1"=sample(rownames(ods), 3),
                        "sample_2"=sample(rownames(ods), 8),
                        "sample_6"=sample(rownames(ods), 5))
genesOfInterest
ods <- OUTRIDER(ods, subsets=list("exampleSubset"=genesOfInterest))
padj(ods, subsetName="exampleSubset")[1:10,1:10]
res <- results(ods, all=TRUE)
res
```

---

## OutriderDataSet-class *OutriderDataSet class and constructors*

---

## Description

The OutriderDataSet class is designed to store the whole OUTRIDER data set needed for an analysis. It is a subclass of `RangedSummarizedExperiment`. All calculated values and results are stored as assays or as annotation in the mcols structure provided by the `RangedSummarizedExperiment` class.

## Usage

```
OutriderDataSet(se, countData, colData, ...)
```

## Arguments

| | |
|---|---|
| `se` | A RangedSummarizedExperiment object or any object which inherits from it and contains a count matrix as the first element in the assay list. |
| `countData` | A simple count matrix. If dim names are provided, they have to be unique. This is only used if no `se` object is provided. |
| `colData` | Additional to the count data a `DataFrame` can be provided to annotate the samples. |
| `...` | Further arguments can be passed to `DESeqDataSet`, which is used to parse the user input and create the initial `RangedSummarizedExperiment` object. |

## Value

An OutriderDataSet object.

## Author(s)

Christian Mertes <mertes@in.tum.de>, Felix Brechtmann <brechtma@in.tum.de>

## Examples

```
ods <- makeExampleOutriderDataSet()
ods

ods <- makeExampleOutriderDataSet(dataset="Kremer")
ods
```

---

`plotAberrantPerSample` *Visualization functions for OUTRIDER*

---

## Description

The OUTRIDER package provides mutliple functions to visualize the data and the results of a full data set analysis.

This is the list of all plotting function provided by OUTRIDER:

- plotAberrantPerSample()
- plotVolcano()
- plotExpressionRank()

- plotQQ()
- plotExpectedVsObservedCounts()
- plotCountCorHeatmap()
- plotCountGeneSampleHeatmap()
- plotSizeFactors()
- plotFPKM()
- plotExpressedGenes()
- plotDispEsts()
- plotPowerAnalysis()
- plotEncDimSearch()

For a detailed description of each plot function please see the details. Most of the functions share the same parameters.

## Usage

```
plotAberrantPerSample(object, ...)

plotCountCorHeatmap(object, ...)

plotManhattan(object, ...)

plotEncDimSearch(object, ...)

plotQQ(object, ...)

plotVolcano(object, ...)

## S4 method for signature 'OutriderDataSet'
plotVolcano(
  object,
  sampleID,
  main,
  padjCutoff = 0.05,
  zScoreCutoff = 0,
  label = "aberrant",
  xaxis = c("zscore", "log2fc", "fc"),
  pch = 16,
  basePlot = FALSE,
  col = c("gray", "firebrick"),
  subsetName = NULL
)

## S4 method for signature 'OutriderDataSet'
plotQQ(
  object,
  geneID,
  main,
  global = FALSE,
  padjCutoff = 0.05,
```

```
  zScoreCutoff = 0,
  samplePoints = TRUE,
  legendPos = "topleft",
  outlierRatio = 0.001,
  conf.alpha = 0.05,
  subsetName = NULL,
  pch = 16,
  xlim = NULL,
  ylim = NULL,
  col = NULL
)

plotExpectedVsObservedCounts(
  ods,
  geneID,
  main,
  basePlot = FALSE,
  log = TRUE,
  groups = c(),
  groupColSet = "Set1",
  label = "aberrant",
  subsetName = NULL,
  ...
)

plotExpressionRank(
  ods,
  geneID,
  main,
  padjCutoff = 0.05,
  zScoreCutoff = 0,
  normalized = TRUE,
  basePlot = FALSE,
  log = TRUE,
  col = c("gray", "firebrick"),
  groups = c(),
  groupColSet = "Accent",
  label = "aberrant",
  subsetName = NULL
)

## S4 method for signature 'OutriderDataSet'
plotCountCorHeatmap(
  object,
  normalized = TRUE,
  rowCentered = TRUE,
  rowGroups = NA,
  rowColSet = NA,
  colGroups = NA,
  colColSet = NA,
  nRowCluster = 4,
  nColCluster = 4,
```

```
  main = "Count correlation heatmap",
  basePlot = TRUE,
  nBreaks = 50,
  show_names = c("none", "row", "col", "both"),
  ...
)

plotCountGeneSampleHeatmap(
  ods,
  normalized = TRUE,
  rowCentered = TRUE,
  rowGroups = NA,
  rowColSet = NA,
  colGroups = NA,
  colColSet = NA,
  nRowCluster = 4,
  nColCluster = 4,
  main = "Count Gene vs Sample Heatmap",
  bcvQuantile = 0.9,
  show_names = c("none", "col", "row", "both"),
  nGenes = 500,
  nBreaks = 50,
  ...
)

## S4 method for signature 'OutriderDataSet'
plotAberrantPerSample(
  object,
  main = "Aberrant Genes per Sample",
  outlierRatio = 0.001,
  col = "Dark2",
  yadjust = 1.2,
  ylab = "Aberrantly expressed genes",
  subsetName = NULL,
  ...
)

plotFPKM(ods, bins = 100)

## S4 method for signature 'OutriderDataSet'
plotDispEsts(
  object,
  compareDisp,
  xlim,
  ylim,
  main = "Dispersion estimates versus mean expression",
  ...
)

plotPowerAnalysis(ods)

## S4 method for signature 'OutriderDataSet'
```

```
plotEncDimSearch(object)

plotExpressedGenes(ods, main = "Statistics of expressed genes")

plotSizeFactors(ods, basePlot = TRUE)

## S4 method for signature 'OutriderDataSet'
plotManhattan(
  object,
  sampleID,
  value = "pvalue",
  chr = NULL,
  main = paste0("Sample: ", sampleID),
  featureRanges = rowRanges(object),
  subsetName = NULL,
  chrColor = c("black", "darkgrey"),
  padjCutoff = 0.05,
  zScoreCutoff = 0,
  highlight.label.size = 5
)
```

## Arguments

| | |
|---|---|
| ... | Additional parameters passed to plot() or plot_ly() if not stated otherwise in the details for each plot function |
| sampleID, geneID | A sample or gene ID, which should be plotted. Can also be a vector. Integers are treated as indices. |
| main | Title for the plot, if missing a default title will be used. |
| padjCutoff, zScoreCutoff | Significance or Z-score cutoff to mark outliers |
| label | Indicates which genes or samples should be labeled. By default all aberrant genes/samples are labelled. Can be set to NULL for no labels. Provide a vector of geneIDs/sampleIDs to label specific genes/samples. |
| xaxis | Indicates which assay should be shown on the x-axis of the volcano plot. Defaults to 'zscore'. Other options are 'fc' and 'log2fc' for the fold-change or log2 fold-change. |
| pch | Integer or character to be used for plotting the points |
| basePlot | if TRUE, use the R base plot version, else use the plotly framework, which is the default |
| col | Set color for the points. If set, it must be a character vector of length 2. (1. normal point; 2. outlier point) or a single character referring to a color palette from RColorBrewer. |
| subsetName | The name of a subset of genes of interest for which FDR corrected pvalues were previously computed. Those FDR values on the subset will then be used to determine aberrant status. Default is NULL (using transcriptome-wide FDR corrected pvalues). |
| global | Flag to plot a global Q-Q plot, default FALSE |
| samplePoints | Sample points for Q-Q plot, defaults to max 30k points |

| | |
|---|---|
| legendPos | Set legendpos, by default topleft. |
| outlierRatio | The fraction to be used for the outlier sample filtering |
| conf.alpha | If set, a confidence interval is plotted, defaults to 0.05 |
| xlim, ylim | The x/y limits for the plot or NULL to use the full data range |
| ods, object | An OutriderDataSet object. |
| log | If TRUE, the default, counts are plotted in log10. |
| groups | A character vector containing either group assignments of samples or sample IDs. Is empty by default. If group assignments are given, the vector must have the same length as the number of samples. If sample IDs are provided the assignment will result in a binary group assignemt. |
| groupColSet | A color set from RColorBrewer or a manual vector of colors, which length must match the number of categories from groups. |
| normalized | If TRUE, the normalized counts are used, the default, otherwise the raw counts |
| rowCentered | If TRUE, the counts are row-wise (gene-wise) centered |
| rowGroups, colGroups | A vector of co-factors (colnames of colData) for color coding the rows. It also accepts a data.frame of dim = (#samples, #groups). Must have more than 2 groups. |
| rowColSet, colColSet | A color set from RColorBrewer/colorRampPalette |
| nRowCluster, nColCluster | Number of clusters to show in the row and column dendrograms. If this argument is set the resulting cluster assignments are added to the OutriderDataSet. |
| nBreaks | number of breaks for the heatmap color scheme. Default to 50. |
| show_names | character string indicating whether to show 'none', 'row', 'col', or 'both' names on the heatmap axes. |
| bcvQuantile | quantile for choosing the cutoff for the biological coefficient of variation (BCV) |
| nGenes | upper limit of number of genes (defaults to 500). Subsets the top n genes based on the BCV. |
| yadjust | Option to adjust position of Median and 90 percentile labels. |
| ylab | The y axis label |
| bins | Number of bins used in the histogram. Defaults to 100. |
| compareDisp | If TRUE, the default, and if the autoCorrect normalization was used it computes the dispersion without autoCorrect and plots it for comparison. |
| value | Indicates which assay is shown in the manhattan plot. Defaults to 'pvalue'. Other options are 'zScore' and 'log2fc'. |
| chr | The chromosomes to be displayed in the plotManhattan function. Default is NULL, i.e. all chromosomes are shown. |
| featureRanges | A GRanges object of the same length as the OutriderDataSet object that contains the genomic positions of features that are shown in the manhattan plot. |
| chrColor | A vector of length 2 giving the two colors used for coloring alternating chromosomes in the manhattan plot. Default colors are 'black' and 'darkgrey'. |
| highlight.label.size | Size of the label of the highlighted genes to be displayed in the plotManhattan. Default value is 5. |

## Details

`plotAberrantPerSample`: The number of aberrant events per sample are plotted sorted by rank. The ... parameters are passed on to the `aberrant` function.

`plotVolcano`: the volcano plot is sample-centric. It plots for a given sample the negative log10 nominal P-values against the Z-scores for all genes.

`plotExpressionRank`: This function plots for a given gene the expression level against the expression rank for all samples. This can be used with normalized and unnormalized expression values.

`plotQQ`: the quantile-quantile plot for a given gene or if `global` is set to `TRUE` over the full data set. Here the observed P-values are plotted against the expected ones in the negative log10 space.

`plotExpectedVsObservedCounts`: A scatter plot of the observed counts against the predicted expression for a given gene.

`plotCountCorHeatmap`: The correlation heatmap of the count data of the full data set. Default the values are log transformed and row centered. This function returns an OutriderDataSet with annotated clusters if requested. The ... arguments are passed to the `pheatmap` function.

`plotCountGeneSampleHeatmap`: A gene x sample heatmap of the raw or normalized counts. By default they are log transformed and row centered. Only the top 500 viable genes based on the BCV (biological coefficient of variation) is used by default.

`plotSizeFactors`: The sizefactor distribution within the dataset.

`plotFPKM`: The distribution of FPKM values. If the OutriderDataSet object contains the `passedFilter` column, it will plot both FPKM distributions for the expressed genes and for the filtered genes.

`plotExpressedGenes`: A summary statistic plot on the number of genes expressed within this dataset. It plots the sample rank (based on the number of expressed genes) against the accumulated statistics up to the given sample.

`plotDispEsts`: Plots the dispersion of the OutriderDataSet model against the normalized mean count. If autoCorrect is used it will also estimate the dispersion without normalization for comparison.

`plotPowerAnalysis`: The power analysis plot should give the user a ruff estimate of the events one can be detected with OUTRIDER. Based on the dispersion of the provided OUTRIDER data set the theoretical P-value over the mean expression is plotted. This is done for different expression levels. The curves are smooths to make the reading of the plot easier.

`plotEncDimSearch`: Visualization of the estimation of the optimal encoding dimension. If Optimal Hard Thresholding was used, the singular values are plotted against their rank. If a hyperparameter optimization was performed, the encoding dimension against the achieved loss (area under the precision-recall curve) is plotted.

`plotManhattan`: Visualizes different metrics for each gene (pvalue, log2 fold-change, z-score) along with the genomic coordinates of the respective gene as a manhattan plot. Detected outlier genes are highlighted in red.

## Value

If base R graphics are used nothing is returned else the plotly or the gplot object is returned.

## Examples

```
ods <- makeExampleOutriderDataSet(dataset="Kremer")
implementation <- 'autoencoder'


mcols(ods)$basepairs <- 300 # assign pseudo gene length for filtering
```

```
ods <- filterExpression(ods)
# restrict FDR correction to set of genes of interest per sample
genesOfInterest <- list(MUC1372 = c("ATPIF1", "UROD", "YBX1",
                                    sample(rownames(ods), 25)),
                        MUC1360 = sample(rownames(ods), 50),
                        MUC1350 = sample(rownames(ods), 75),
                        X76619 = sample(rownames(ods), 20),
                        X74172 = sample(rownames(ods), 150))
ods <- OUTRIDER(ods, implementation=implementation, subsets=list("exampleGenes"=genesOfInterest))

plotAberrantPerSample(ods)
plotAberrantPerSample(ods, subsetName="exampleGenes")

plotVolcano(ods, 49)
plotVolcano(ods, 'MUC1365', basePlot=TRUE)
plotVolcano(ods, 'MUC1351', basePlot=TRUE, xaxis="log2fc", label=c("NBPF16"))
plotVolcano(ods, 'MUC1372', basePlot=TRUE, subsetName="exampleGenes")

plotExpressionRank(ods, 35)
plotExpressionRank(ods, 35, subsetName="exampleGenes")
plotExpressionRank(ods, "NDUFS5", normalized=FALSE, label="MUC1372",
    log=FALSE, main="Over expression outlier", basePlot=TRUE)

plotQQ(ods, 149)
plotQQ(ods, 149, subsetName="exampleGenes")
plotQQ(ods, global=TRUE, outlierRatio=0.001)

plotExpectedVsObservedCounts(ods, 149)
plotExpectedVsObservedCounts(ods, "ATAD3C", basePlot=TRUE)
plotExpectedVsObservedCounts(ods, "UROD", subsetName="exampleGenes")

plotExpressedGenes(ods)

sex <- sample(c("female", "male"), dim(ods)[2], replace=TRUE)
colData(ods)$Sex <- sex
ods <- plotCountCorHeatmap(ods, nColCluster=4, normalized=FALSE)
ods <- plotCountCorHeatmap(ods, colGroup="Sex", colColSet="Set1")
table(colData(ods)$clusterNumber_4)

plotCountGeneSampleHeatmap(ods, normalized=FALSE)
plotCountGeneSampleHeatmap(ods, rowGroups="theta",
        rowColSet=list(c("white", "darkgreen")))

plotSizeFactors(ods)

mcols(ods)$basepairs <- 1
mcols(ods)$passedFilter <- rowMeans(counts(ods)) > 10
plotFPKM(ods)

plotDispEsts(ods, compareDisp=FALSE)

plotPowerAnalysis(ods)

## Not run:
# for speed reasons we only search for 5 different dimensions
ods <- estimateBestQ(ods, useOHT=FALSE, params=c(3, 10, 20, 35, 50),
        implementation=implementation)
```

```
plotEncDimSearch(ods)

## End(Not run)

# To show the pvalues of a sample in a manhattan plot, rowRanges(ods) must
# contain the genomic position of each feature or a GRanges object must
# be provided
## Not run:
# in case rowRanges(ods) is a GRangesList, run this first once to speed up:
rowRanges(ods) <- unlist(endoapply(rowRanges(ods), range))

## End(Not run)
gr <- GRanges(
         seqnames=sample(paste0("chr", 1:22), nrow(ods), replace=TRUE),
         ranges=IRanges(start=runif(nrow(ods), min=0, max=1e5), width=100))
plotManhattan(ods, "MUC1350", value="pvalue", featureRanges=gr)
plotManhattan(ods, "MUC1350", value="l2fc", featureRanges=gr)
plotManhattan(ods, "MUC1372", featureRanges=gr, subsetName="exampleGenes")
```

---

`results` *Accessor function for the 'results' object in an OutriderDataSet object.*

---

## Description

This function assembles a results table of significant outlier events based on the given filter criteria. The table contains various information accumulated over the analysis pipeline.

## Usage

```
results(object, ...)

## S4 method for signature 'OutriderDataSet'
results(
  object,
  padjCutoff = 0.05,
  zScoreCutoff = 0,
  round = 2,
  all = FALSE,
  returnTranscriptomewideResults = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | An OutriderDataSet object |
| `...` | Additional arguments, currently not used |
| `padjCutoff` | The significant threshold to be applied |
| `zScoreCutoff` | If provided additionally a z score threshold is applied |
| `round` | Can be TRUE, defaults to 2, or an integer used for rounding with `round` to make the output more user friendly |

`all` By default FALSE, only significant read counts are listed in the results. If TRUE all results are assembled resulting in a data.table of length samples x genes.

`returnTranscriptomewideResults`
If FDR corrected pvalues for subsets of genes of interest have been calculated, this parameter indicates whether additionally the transcriptome-wide results should be returned as well (default), or whether only results for those subsets should be retrieved.

## Value

A data.table where each row is an outlier event and the columns contain additional information about this event. In details the table contains:

`sampleID/geneID`
The gene or sample ID as provided by the user, e.g. `rowData(ods)` and `colData(ods)`, respectively.

`pValue/padjust` The nominal P-value and the FDR corrected P-value (transcriptome-wide) indicating the outlier status.

`zScore/l2fc` The z score and $\log_2$ fold change as computed by `computeZscores`.

`rawcounts` The observed read counts.

`normcounts` The expected count given the fitted autoencoder model for the given gene-sample combination.

`meanRawcounts/meanCorrected`
For this gene, the mean of the observed or expected counts, respectively, given the fitted autoencoder model.

`theta` The dispersion parameter of the NB distribution for the given gene.

`aberrant` The transcriptome-wide outlier status of this event: `TRUE` or `FALSE`.

`AberrantBySample/AberrantByGene`
Number of outliers for the given sample or gene (transcriptome-wide), respectively.

`padj_rank` Rank of this outlier event within the given sample.

`padjust_FDRset` The FDR corrected P-value with respect to the gene subset called 'FDRset', if gene subsets were specified during the P-value computation. Find more details at `computePvalues`.

## Examples

```
ods <- makeExampleOutriderDataSet()

ods <- OUTRIDER(ods)

res <- results(ods, all=TRUE)
res

# example of retrieving results with FDR correction limited to a
# set of genes of interest
genesOfInterest <- list("sample_1"=sample(rownames(ods), 3),
                        "sample_2"=sample(rownames(ods), 8),
                        "sample_6"=sample(rownames(ods), 5))
genesOfInterest
ods <- computePvalues(ods, subsets=list("exampleSubset"=genesOfInterest))
res <- results(ods, all=TRUE, returnTranscriptomewideResults=FALSE)
```

```
res
```

---

`sampleExclusionMask` *Sample exclusion*

---

## Description

To exclude a sample from the fit process, one can use this function to mask specific samples. This can be used if replicates are present.

## Usage

```
sampleExclusionMask(ods, aeMatrix = FALSE)

sampleExclusionMask(ods) <- value
```

## Arguments

| | |
|---|---|
| `ods` | An OutriderDataSet object |
| `aeMatrix` | If `TRUE`, it returns a 0/1 matrix for the internal autoencoder functions in the form of feature x sample |
| `value` | A logical vector of the length of the samples. If `TRUE`, the corresponding sample will be excluded from the autoencoder fit. |

## Value

The exclusion vector/matrix.

## Examples

```
ods <- makeExampleOutriderDataSet()
sampleExclusionMask(ods) <- sample(c(FALSE, TRUE), ncol(ods), replace=TRUE)

sampleExclusionMask(ods)
```

---

`sizeFactors` *SizeFactors accessor and estimation function*

---

## Description

Accessor functions for the 'sizeFactors' information in a OutriderDataSet object.

## Usage

```
## S4 method for signature 'OutriderDataSet'
sizeFactors(object)

## S4 replacement method for signature 'OutriderDataSet,numeric'
sizeFactors(object) <- value

## S4 replacement method for signature 'OutriderDataSet,NULL'
sizeFactors(object) <- value

## S4 method for signature 'OutriderDataSet'
estimateSizeFactors(object)
```

## Arguments

| | |
|---|---|
| object | OutriderDataSet |
| value | A numberic vector of sizeFactors |

## Details

The estimation of the size factors can also make use of the existing log geometric means in the object. Those can be loaded from an existing model.

## Value

An OutriderDatasSet with the estimated sizeFactors, or with the getter function it returns a numeric vector containing the sizeFactors.

## See Also

estimateSizeFactors

## Examples

```
ods <- makeExampleOutriderDataSet()
ods <- estimateSizeFactors(ods)
head(sizeFactors(ods))

sizeFactors(ods) <- runif(dim(ods)[2], 0.5, 1.5)
sizeFactors(ods)
counts(ods, normalized=TRUE)[1:10,1:10]
```

# Index


‘sampleExclusionMask<-‘
(sampleExclusionMask), 28

aberrant, 3, *24*
aberrant,OutriderDataSet-method
(aberrant), 3

BiocParallelParam, *7*, *17*

computeGeneLength, 4
computeLatentSpace, 4
computePvalues, 5, *16*, *27*
computePvalues,OutriderDataSet-method
(computePvalues), 5
computeZscores, 6, *16*, *27*
computeZscores,OutriderDataSet-method
(computeZscores), 6
controlForConfounders, 7, *15, 16*
counts, 8
counts,OutriderDataSet-method (counts),
8
counts<-,OutriderDataSet,matrix-method
(counts), 8

DESeqDataSet, *18*
dispersion, (getBestQ), 13
dispersions (getBestQ), 13
dispersions,OutriderDataSet-method
(getBestQ), 13

estimateBestQ, 9
estimateBestQ, (estimateBestQ), 9
estimateDispersions, *13*
estimateSizeFactors, *16*, *29*
estimateSizeFactors (sizeFactors), 28
estimateSizeFactors,OutriderDataSet-method
(sizeFactors), 28

filterExpression, 10
filterExpression,OutriderDataSet-method
(filterExpression), 10
findInjectZscore (estimateBestQ), 9
fit, 12, *16*
fpkm, *10*, *12*, 12
fpm, *12*
fpm (fpkm), 12

getBestQ, 13
getter_setter_functions (getBestQ), 13

makeExampleOutriderDataSet, 14

normalizationFactors, *8*, 15, *16*
normalizationFactors,OutriderDataSet-method
(normalizationFactors), 15
normalizationFactors<-,OutriderDataSet,data.frame-metho
(normalizationFactors), 15
normalizationFactors<-,OutriderDataSet,DataFrame-method
(normalizationFactors), 15
normalizationFactors<-,OutriderDataSet,matrix-method
(normalizationFactors), 15
normalizationFactors<-,OutriderDataSet,NULL-method
(normalizationFactors), 15

OUTRIDER, 16
OutriderDataSet, *8*, *15*
OutriderDataSet
(OutriderDataSet-class), 17
OutriderDataSet-class, 17

padj (getBestQ), 13
padj, (getBestQ), 13
pheatmap, *24*
plotAberrantPerSample, 18
plotAberrantPerSample,OutriderDataSet-method
(plotAberrantPerSample), 18
plotCountCorHeatmap
(plotAberrantPerSample), 18
plotCountCorHeatmap,OutriderDataSet-method
(plotAberrantPerSample), 18
plotCountGeneSampleHeatmap
(plotAberrantPerSample), 18
plotDispEsts (plotAberrantPerSample), 18
plotDispEsts,OutriderDataSet-method
(plotAberrantPerSample), 18
plotEncDimSearch
(plotAberrantPerSample), 18
plotEncDimSearch,OutriderDataSet-method
(plotAberrantPerSample), 18

plotExpectedVsObservedCounts
(plotAberrantPerSample), 18
plotExpressedGenes
(plotAberrantPerSample), 18
plotExpressionRank
(plotAberrantPerSample), 18
plotFPKM (plotAberrantPerSample), 18
plotFunction (plotAberrantPerSample), 18
plotFunctions (plotAberrantPerSample),
18
plotManhattan (plotAberrantPerSample),
18
plotManhattan,OutriderDataSet-method
(plotAberrantPerSample), 18
plotPowerAnalysis
(plotAberrantPerSample), 18
plotQQ (plotAberrantPerSample), 18
plotQQ,OutriderDataSet-method
(plotAberrantPerSample), 18
plotSizeFactors
(plotAberrantPerSample), 18
plotVolcano (plotAberrantPerSample), 18
plotVolcano,OutriderDataSet-method
(plotAberrantPerSample), 18
pValue (getBestQ), 13
pValue, (getBestQ), 13

results, 26
results,OutriderDataSet-method
(results), 26
round, *26*

sampleExclusionMask, 28
sampleExclusionMask,
(sampleExclusionMask), 28
sampleExclusionMask<-
(sampleExclusionMask), 28
sizeFactors, *8*, *16*, 28
sizeFactors,OutriderDataSet-method
(sizeFactors), 28
sizeFactors<- (sizeFactors), 28
sizeFactors<-,OutriderDataSet,NULL-method
(sizeFactors), 28
sizeFactors<-,OutriderDataSet,numeric-method
(sizeFactors), 28

theta (getBestQ), 13
theta, (getBestQ), 13

zScore (getBestQ), 13
zScore, (getBestQ), 13