

# Package ‘qsea’

January 21, 2025

**Type** Package

**Title** IP-seq data analysis and vizualization

**Version** 1.32.0

**Date** 2023-03-22

**Description** qsea (quantitative sequencing enrichment analysis) was developed as the successor of the MEDIPS package for analyzing data derived from methylated DNA immunoprecipitation (MeDIP) experiments followed by sequencing (MeDIP-seq). However, qsea provides several functionalities for the analysis of other kinds of quantitative sequencing data (e.g. ChIP-seq, MBD-seq, CMS-seq and others) including calculation of differential enrichment between groups of samples.

**License** GPL-2

**biocViews** Sequencing, DNAMethylation, CpGIsland, ChIPSeq, Preprocessing, Normalization, QualityControl, Visualization, CopyNumberVariation, ChipOnChip, DifferentialMethylation

**Depends** R (>= 4.3)

**Imports** Biostrings, graphics, gtools, methods, stats, utils, HMMcopy, rtracklayer, BSgenome, GenomicRanges, Rsamtools, IRanges, limma, GenomeInfoDb, BiocGenerics, grDevices, zoo, BiocParallel, S4Vectors

**VignetteBuilder** knitr

**Suggests** BSgenome.Hsapiens.UCSC.hg19, MEDIPSData, testthat, BiocStyle, knitr, rmarkdown, BiocManager, MASS

**ByteCompile** no

**NeedsCompilation** yes

**git\_url** <https://git.bioconductor.org/packages/qsea>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** cb91f8f

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2025-01-20

**Author** Matthias Lienhard [aut, cre] (<<https://orcid.org/0000-0002-2549-3142>>),  
Lukas Chavez [aut] (<<https://orcid.org/0000-0002-8718-8848>>),  
Ralf Herwig [aut] (<<https://orcid.org/0000-0002-9335-1760>>)

**Maintainer** Matthias Lienhard <lienhard@molgen.mpg.de>

## Contents

|                         |           |
|-------------------------|-----------|
| qsea-package            | 2         |
| addCNV                  | 3         |
| addContrast             | 4         |
| addCoverage             | 5         |
| addEnrichmentParameters | 6         |
| addLibraryFactors       | 7         |
| addNewSamples           | 8         |
| addOffset               | 9         |
| addPatternDensity       | 10        |
| addSeqPref              | 11        |
| createQseaSet           | 12        |
| fitNBglm                | 13        |
| getExampleQseaSet       | 14        |
| getPCA                  | 15        |
| isSignificant           | 16        |
| makeTable               | 17        |
| normMethod              | 19        |
| plotCNV                 | 21        |
| plotCoverage            | 22        |
| plotEnrichmentProfile   | 23        |
| plotPCA                 | 25        |
| qseaGLM-class           | 26        |
| qseaPCA-class           | 26        |
| qseaSet-class           | 27        |
| regionStats             | 28        |
| <b>Index</b>            | <b>30</b> |

---

qsea-package

*QSEA: Quantitative sequencing enrichment analysis and visualization*


---

### Description

QSEA (quantitative sequencing enrichment analysis) was developed as the successor of the MEDIPS package for analyzing data derived from methylated DNA immunoprecipitation (MeDIP) experiments followed by sequencing (MeDIP-seq). However, qsea provides functionality for the analysis of other kinds of quantitative sequencing data (e.g. ChIP-seq, MBD-seq, CMS-seq and others) including calculation of differential enrichment between groups of samples.

### Author(s)

Matthias Lienhard, Lukas Chavez and Ralf Herwig

Maintainer: Matthias Lienhard <lienhard@molgen.mpg.de>

### References

Lienhard M, Grimm C, Morkel M, Herwig R, Chavez L., (Bioinformatics, 2014): MEDIPS: genome-wide differential coverage analysis of sequencing data derived from DNA enrichment experiments.

---

addCNV *estimate CNV information and add to qseaSet object*

---

### Description

This function adds information on Copy Number Variation (CNV) to the qseaSet object, which is used for normalization. Sample wise CNV information can either be provided, or estimated from input or enrichment sequencing data, by incorporating functions of the HMMcopy package.

### Usage

```
addCNV(qs, file_name, window_size=1000000, paired=FALSE, fragment_length, cnv,
mu=log2(c(1/2, 2/3, 1, 3/2, 2, 3)), normal_idx, plot_dir, MeDIP=FALSE,
parallel=FALSE)
```

### Arguments

|                 |  |
|-----------------|--|
| qs              | the qseaSet object   |
| cnv             | pre-computed CNV information for each sample. If provided, the following parameters are ignored  |
| file_name       | column name of the sample table for the sequencing files, from which CNV information are computed  |
| window_size     | window size for CNV analysis   |
| paired          | are files in file_name column paired end   |
| fragment_length | for single end sequencing, provide the average fragment length   |
| mu              | a priori CNV levels of different states, parameter passed to HMMcopy   |
| normal_idx      | index of samples which are assumed to be CNV free. The median of these samples serves as "normal" CNV reference level, and CNV are computed relative to this reference level. By default, QSEA looks for samples with "normal" or "control" in its name.   |
| plot_dir        | If provided, detail CNV plots for each chromosome and each sample are created in the provided directory  |
| MeDIP           | If set TRUE, QSEA assumes that provided files are methylation enriched sequencing data. In this case, only fragments without CpG dinucleotides are considered. This option allows QSEA to infer CNV information from MeDIP or MDB seq experiments directly |
| parallel        | Switch for parallel computing, using BiocParallel  |

### Value

The qseaSet object, extended by the CNV information

### Author(s)

Mathias Lienhard

**See Also**

HMMsegment

**Examples**

```
library("BSgenome.Hsapiens.UCSC.hg19")

bam_hESCs_1 = system.file("extdata",
  "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")
bam_hESCs_2 = system.file("extdata",
  "hESCs.MeDIP.Rep2.chr22.bam", package="MEDIPSData")
sample_table=data.frame(sample_name=paste0("hESCs_", 1:2),
  file_name=c(bam_hESCs_1,bam_hESCs_2),
  group=rep("hESC",2), stringsAsFactors=FALSE)
qseaSet=createQseaSet(sampleTable=sample_table,
  BSgenome="BSgenome.Hsapiens.UCSC.hg19",
  chr.select="chr22",
  window_size=500)

#this is an example for computing CNVs from MeDIP data. A very limited example
#however, since the samples do not contain CNVs.
qseaSet=addCNV(qseaSet, fragment_length=300, file_name="file_name", MeDIP=TRUE,
  window_size=1000000)
```

addContrast

*fit GLMs to reduced model and test for significance***Description**

This function fits negative binomial GLMs to reduced models defined either by the "contrast" parameter, or by one or several model coefficients (specified by "coef" parameter) set to zero. Subsequently, a likelihood ratio test is applied, to identify windows significantly dependent on the tested coefficient.

**Usage**

```
addContrast(qs,glm,contrast,coef,name,verbose=TRUE, nChunks = NULL,
  parallel = FALSE )
```

**Arguments**

|          |  |
|----------|--|
| qs       | a qseaSet object   |
| glm      | a qseaGLM object   |
| contrast | numeric vector specifying a contrast of the model coefficients. This contrast can for example be defined using <code>limma::makeContrasts()</code> |
| coef     | alternatively defines the contrast by coefficient(s) of the model tested to be equal to zero.  |
| name     | short descriptive name for the contrast (as "TvsN"), used for examples in columns of result tables   |

|          |   |
|----------|---|
| verbose  | more messages that document the process |
| nChunks  | fit GLMs in multiple chunks             |
| parallel | use multicore processing                |

**Value**

This function returns the qseaGLM object, extended by the fitted coefficients of the reduced GLMs, as well as the test statistics. Note that one qseaGLM object can contain several contrasts.

**Author(s)**

Mathias Lienhard

**See Also**

limma::makeContrasts(), fitNBglm(), isSignificant()

**Examples**

```
qs=getExampleQseaSet()
design=model.matrix(~group, getSampleTable(qs))
TvN_glm=fitNBglm(qs, design, norm_method="beta")
TvN_glm=addContrast(qs,TvN_glm, coef=2, name="TvN")
```

---

|             |                               |
|-------------|-------------------------------|
| addCoverage | <i>Import sequencing data</i> |
|-------------|-------------------------------|

---

**Description**

This function imports the alignment files (in sam/bam format) and counts the reads per genomic window or directly imports coverage files (in wiggle/bigwiggle format)

**Usage**

```
addCoverage(qs, fragment_length, uniquePos=TRUE, minMapQual=1, paired=FALSE,
parallel=FALSE)
```

**Arguments**

|                 |  |
|-----------------|--|
| qs              | qseaSet object, e.g. produced by the createQseaSet() function  |
| fragment_length | For single end data, provide the expected fragment length  |
| paired          | If set to TRUE, data is considered to be paired end sequencing, and the actual fragments size is used.                               |
| uniquePos       | If set to TRUE, fragments with same position and orientation are considered to be PCR duplicates and replaced by one representative. |
| minMapQual      | The minimal mapping quality for reads to be considered. Note that the definition of mapping quality depends on the alignment tool.   |
| parallel        | Switch for parallel computing, using BiocParallel  |

## Details

The coverage is imported from the files specified in the `file_name` column of the sample table, provided for the `createQseaSet()` function. In case of alignment files, the reads are counted for the window at the center of the sequencing fragment. For single end data, Filetypes is detected automatically from the file suffix.

## Value

The function returns the `qseaSet` object, extended by the number of reads per window for all samples

## Author(s)

Mathias Lienhard

## See Also

`crateQseaSet`

## Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")

bam_hESCs_1 = system.file("extdata",
  "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")
bam_hESCs_2 = system.file("extdata",
  "hESCs.MeDIP.Rep2.chr22.bam", package="MEDIPSData")
sample_table=data.frame(sample_name=paste0("hESCs_", 1:2),
  file_name=c(bam_hESCs_1,bam_hESCs_2),
  group=rep("hESC",2), stringsAsFactors=FALSE)
qseaSet=createQseaSet(sampleTable=sample_table,
  BSgenome="BSgenome.Hsapiens.UCSC.hg19",
  chr.select="chr22",
  window_size=500)
qseaSet=addCoverage(qseaSet, fragment_length=300)
```

---

addEnrichmentParameters

*Enrichment analysis*

---

## Description

This function analyses the dependency of enrichment on a sequence pattern, based on a subset of windows for which the signal is known.

## Usage

```
addEnrichmentParameters(qs, enrichmentPattern, signal, windowIdx,
  min_wd=5, bins=seq(.5,40.5,1))
```

**Arguments**

|                   |  |
|-------------------|--|
| qs                | The qseaSet object   |
| enrichmentPattern | The name of the pattern, on which the enrichment depends on (usually CpG for methylation analysis). This name must correspond to the name specified in addPatternDensity() |
| windowIdx         | vector of window indices, for which "true" values are known (or can be estimated)  |
| signal            | Matrix containing the known (or estimated) values for all samples and all specified windows, as a numeric matrix. These values are expected to be between 0 and 1.         |
| bins              | For the enrichment analysis, windows are binned according to pattern density. This parameter specifies the bins.   |
| min_wd            | minimal number of windows per bin to be considered   |

**Value**

The function returns the qseaSet object, extended by the parameters of the enrichment profiles for all samples

**Author(s)**

Mathias Lienhard

**See Also**

plotEnrichmentProfile, addPatternDensity

**Examples**

```
qs=getExampleQseaSet(enrichmentAnalysis=FALSE)
#this procedure assumes that regions with low CpG density is 80% methylated
#on average, and regions within CpG islands are 25% methylated on average.
wd=which(getRegions(qs)$CpG_density>1 &
  getRegions(qs)$CpG_density<15)
signal=(15-getRegions(qs)$CpG_density[wd])*0.55/15+0.25
signal=matrix(signal,nrow=length(signal),ncol=length(getSampleNames(qs)))
qs=addEnrichmentParameters(qs, enrichmentPattern="CpG",
  windowIdx=wd, signal=signal)
```

---

addLibraryFactors      *Estimate effective library size*

---

**Description**

Normalization factors for effective library size are computed using the trimmed mean of m-values approach (TMM).

**Usage**

```
addLibraryFactors(qs, factors,...)
```

**Arguments**

|         |   |
|---------|---|
| qs      | The qseaSet object  |
| factors | In case normalization factors have been pre-computed by the user, they can be passed with this parameter. In this case QSEA adds this factors to the qseaSet object and does not compute normalization factors. |
| ...     | Further parameters used for the TMM normalization (see details)   |

**Details**

The user can specify the TMM normalization by setting the following additional parameters, which are passed to the internal functions. `\trimA` [default: `c(.5,.99)`] lower and upper quantiles for trimming of A values `\trimM` [default: `c(.1,.9)`] lower and upper quantiles for trimming of M values `\doWeighting` [default: `TRUE`] computes a weighted TMM `\ref` [default: `1`] the index of the reference sample `\plot` [default: `FALSE`] if set to `TRUE`, MvsA plots depicting the TMM normalization are created. `\nReg` [default: `500000`] Number of regions to be analyzed for normalization. Regions are drawn uniformly over the whole genome.

**Value**

This function returns the qseaSet object, containing effective library size normalization factors.

**Author(s)**

Mathias Lienhard

**See Also**

`edgeR::calcNormFactors`

**Examples**

```
qs=getExampleQseaSet(expSamplingDepth=500*10^(1:5), repl=5)
#in this case, the first sample has only view reads, so it is important to set
#the reference sample
qs=addLibraryFactors(qs, plot=TRUE, ref="Sim5N")
```

---

addNewSamples

*Extends an existing qseaSet by new samples*

---

**Description**

This function allows the qseaSet to be extended by new samples, provided in the sample table.

**Usage**

```
addNewSamples(qs, sampleTable, force=FALSE, parallel=FALSE)
```



**Arguments**

|             |  |
|-------------|--|
| qs          | The qseaSet object to be extended  |
| sampleTable | data.frame, describing the samples. Must be in same format as getSampleTable(qs)                   |
| force       | force adding of new samples, even if existing CNV or enrichment information requires recomputation |
| parallel    | parallel processing of alignment files   |

**Value**

An object of class qseaSet, including the new samples.

**Author(s)**

Mathias Lienhard

**Examples**

```
library("BSgenome.Hsapiens.UCSC.hg19")
data(samplesNSCLC, package="MEDIPSData")
path=system.file("extdata", package="MEDIPSData")
samples_NSCLC$file_name=paste0(path,"/",samples_NSCLC$file_name )
originalQseaSet=createQseaSet(sampleTable=samples_NSCLC[1:4,],
                              BSgenome="BSgenome.Hsapiens.UCSC.hg19", chr.select="chr22",
                              window_size=500)
originalQseaSet=addCoverage(originalQseaSet, uniquePos=TRUE, paired=TRUE)
qseaSet=addNewSamples(originalQseaSet, samples_NSCLC)
```

---

addOffset

*Estimate background reads*

---

**Description**

This function sets the background reads offset parameters for the qseaSet object, either by estimating offset reads, or by setting user provided values.

**Usage**

```
addOffset(qs,enrichmentPattern , maxPatternDensity=0.01,offset)
```

**Arguments**

|                   |   |
|-------------------|---|
| qs                | the qseaSet object  |
| enrichmentPattern | name of the enrichment pattern, as specified in addPatternDensity   |
| maxPatternDensity | Maximum pattern density, at which the window is treated as pattern free.  |
| offset            | This parameter alternatively allows to specify the amount of background reads for each sample manually. In this case, please provide average background reads for CNV free windows in rpkm scale. |

**Value**

The function returns the `qseaSet` object, extended by the estimated amount of background reads for all samples

**Author(s)**

Mathias Lienhard

**See Also**

`addPatternDensity`, `getOffset`

**Examples**

```
#simulate data with varying background fractions
qs=getExampleQseaSet(expSamplingDepth=5e4, repl=5,bgfraction=seq(0,.8,.2))
#estimate the background in simulated data
addOffset(qs, "CpG", maxPatternDensity=0.7)
#return the background on different scales
getOffset(qs, scale="fraction") #estimated fraction of total reads
getOffset(qs, scale="rpw") #average background reads per CNV free window
```

---

`addPatternDensity`      *Infer sequence pattern density values and add to qseaSet object*

---

**Description**

This function estimates the average occurrences of a sequence pattern (such as CpG dinucleotides) within the overlapping sequencing fragments for each genomic window

**Usage**

```
addPatternDensity(qs, pattern,name, fragment_length, fragment_sd,
patternDensity, fixed=TRUE, masks=c("AGAPS","AMB", "RM", "TRF")[1:2])
```

**Arguments**

|                              |   |
|------------------------------|---|
| <code>qs</code>              | a <code>qseaSet</code> object   |
| <code>pattern</code>         | actual sequence of the pattern (e.g. "CG")  |
| <code>,</code>               |   |
| <code>name</code>            | a name for the sequence pattern(e.g. "CpG")   |
| <code>,</code>               |   |
| <code>fragment_length</code> | the average fragment length to be assumed for pattern density estimation. If omitted, this parameter is taken from the <code>qseaSet</code> object.               |
| <code>fragment_sd</code>     | the standard deviation of fragment length to be assumed for pattern density estimation. If omitted, this parameter is taken from the <code>qseaSet</code> object. |

- `patternDensity` this parameter alternatively allows to specify the pattern density manually. In this case, please provide a numerical vector, containing a value (greater than 0) for each genomic window.
- `fixed` if FALSE, an IUPAC ambiguity code in the pattern can match any letter in the reference genome that is associated with the code, and vice versa.
- `masks` names of the BSgenome masks to be active.

### Value

The function returns the `qseaSet` object, extended by the pattern density for all genomic windows

### Author(s)

Mathias Lienhard

### Examples

```
library("BSgenome.Hsapiens.UCSC.hg19")
bam_hESCs_1 = system.file("extdata",
  "hESCs.MeDIP.Rep1.chr22.bam", package="MEDIPSData")
bam_hESCs_2 = system.file("extdata",
  "hESCs.MeDIP.Rep2.chr22.bam", package="MEDIPSData")
sample_table=data.frame(sample_name=paste0("hESCs_", 1:2),
  file_name=c(bam_hESCs_1,bam_hESCs_2),
  group=rep("hESC",2), stringsAsFactors=FALSE)
qseaSet=createQseaSet(sampleTable=sample_table,
  BSgenome="BSgenome.Hsapiens.UCSC.hg19",
  chr.select="chr22",
  window_size=500)
qseaSet=addPatternDensity(qseaSet, "CG", name="CpG", fragment_length=300)
```

---

addSeqPref

*Add sequence preference to qseaSet object*

---

### Description

This function allows to add window specific sequencing preference, that can be used by the normalization procedure. This preference can be defined by the user, or estimated from sequencing of input libraries.

### Usage

```
addSeqPref(qs, seqPref,file_name, fragment_length, paired=FALSE,
  uniquePos=TRUE, alpha=0.05, pseudocount=5, cut=3)
```

**Arguments**

|                 |  |
|-----------------|--|
| qs              | a qseaSet object   |
| seqPref         | A vector with predefined sequencing preference for each window. Values are interpreted as log <sub>2</sub> ratios relative to normal/average sequencing preference.  |
| file_name       | alternatively, the sequencing preference can be estimated from input sequencing. In this case, provide the column of the sample table that contains the file names for input sequencing alignment or coverage files. |
| fragment_length | for single end data, provide the expected fragment length  |
| paired          | if set to TRUE, data is considered to be paired end sequencing, and the actual fragments size is used.   |
| uniquePos       | if set to TRUE, fragments with same position and orientation are considered to be PCR duplicates and replaced by one representative.   |
| alpha           | currently ignored  |
| pseudocount     | this value is added to the coverage of each window, to smooth the estimates.   |
| cut             | absolute log <sub>2</sub> value threshold for windows to be excluded from later analysis due to extreme preference values.   |

**Value**

the function returns the qseaSet object, extended by the sequencing preference for all genomic windows.

**Author(s)**

Mathias Lienhard

---

|               |                                  |
|---------------|----------------------------------|
| createQseaSet | <i>Prepares a qseaSet Object</i> |
|---------------|----------------------------------|

---

**Description**

This method prepares the qseaSet object, and prepares genome wide bins. Coverage and normalization parameters are added in succeeding functions.

**Usage**

```
createQseaSet(sampleTable,BSgenome, chr.select,Regions, window_size=250 )
```

**Arguments**

|            |   |
|------------|---|
| BSgenome   | name of BSgenome package  |
| Regions    | GRanges object. If specified, only selected regions are processed |
| chr.select | If specified, only selected chromosomes are processed             |

|             |  |
|-------------|--|
| sampleTable | data.frame, containing at least 3 columns:<br>the sample names (sample_name), paths to alignment or coverage file in sam/bam/wiggle/bigwig format (file_name), and one or more test condition(s) (group).<br>Optionally it may contain a column with alignment or coverage files for CNV analysis, and further information in the samples that are of interest for the analysis. |
| window_size | size for the genome wide bins in base pairs  |

**Value**

An object of class qseaSet, containing the sample and genome information.

**Author(s)**

Mathias Lienhard

**Examples**

```
library("BSgenome.Hsapiens.UCSC.hg19")
bam_hESCs_1 = system.file("extdata", "hESCs.MeDIP.Rep1.chr22.bam",
  package="MEDIPSData")
bam_hESCs_2 = system.file("extdata", "hESCs.MeDIP.Rep2.chr22.bam",
  package="MEDIPSData")
samplesTable=data.frame(sample_name=paste0("hESCs_", 1:2),
  file_name=c(bam_hESCs_1,bam_hESCs_2),
  group=rep("hESC",2),stringsAsFactors=FALSE)
qs=createQseaSet(samplesTable, BSgenome="BSgenome.Hsapiens.UCSC.hg19",
  chr.select="chr22", window_size=500)
```

---

fitNBglm

*Fit GLM for each window*


---

**Description**

This function fits a negative binomial GLM for each genomic window, according to the design matrix.

**Usage**

```
fitNBglm(qs,design,link="log",keep, disp_method="region_wise",
  norm_method="rpkm",init_disp=0.5 ,verbose=TRUE, minRowSum=10, pseudocount=1,
  disp_iter = 3, nChunks = NULL, parallel = FALSE)
```

**Arguments**

|        |  |
|--------|--|
| qs     | a qseaSet object   |
| design | the design matrix for the GLMs   |
| link   | name of the link function. Currently, only the canonical dQuotelog link function is implemented. |
| keep   | indices of windows to be included in the analysis.   |

|             |  |
|-------------|--|
| disp_method | method to estimate dispersion parameters. Allowed values are dQuote"region_wise" for independent window wise estimates, dQuote"common" for a single estimate for all windows, dQuote"cutAtQuantiles" for window wise estimates trimmed at the 25% and 75% quantiles, or dQuote"initial" for using the dispersion parameters provided with the init_disp parameter. |
| norm_method | normalization method, as defined by normMethod() function  |
| init_disp   | initial estimate for dispersion parameter. Either a single parameter for all regions, or a vector with window wise parameters.   |
| verbose     | more messages that document the process  |
| minRowSum   | filter out windows with less than minRowSum reads over all samples   |
| pseudocount | this value is added to the read counts   |
| disp_iter   | number of iterations for dispersion estimation   |
| nChunks     | fit GLMs in multiple chunks  |
| parallel    | use multicore processing   |

**Value**

This function returns a qseaGLM object, containing the fitted coefficients of the GLMs.

**Author(s)**

Mathias Lienhard

**See Also**

addContrast()

**Examples**

```
#tbd
qs=getExampleQseaSet()
design=model.matrix(~group, getSampleTable(qs))
TvN_glm=fitNBglm(qs, design, norm_method="beta")
```

---

getExampleQseaSet      *Simulation of MeDIP seq QSEA set*

---

**Description**

Creates a example qseaSet object by sampling reads for simulated Tumor and Normal samples. Number of replicates, sequencing depth and fraction of background reads can be specified.

**Usage**

```
getExampleQseaSet(CpG=TRUE, CNV=TRUE, repl=2,
  doSampling=TRUE, enrichmentAnalysis=TRUE, expSamplingDepth=50000,
  bgfraction=.1)
```

**Arguments**

|                    |  |
|--------------------|--|
| CpG                | if TRUE CpG density is added to the object               |
| CNV                | if TRUE CNV are emulated for the tumor samples           |
| repl               | number of replicates for tumor and normal samples        |
| doSampling         | if TRUE, read counts are sampled and added to the object |
| enrichmentAnalysis | if TRUE, parameters for enrichment profiles are added    |
| expSamplingDepth   | expected value of sequencing depth                       |
| bgfraction         | fraction of background reads                             |

**Details**

The function creates an example and test qseaSet object for an toy example genome (one chromosome, 50kb) with 500 bases windows.

**Value**

The qseaSet object

**Author(s)**

Mathias Lienhard

**See Also**

createQseaSet()

**Examples**

```
qs=getExampleQseaSet()
```

---

getPCA

*Principle Component Analysis (PCA) in QSea*

---

**Description**

The getPCA() function performs a Principle Component Analysis (PCA) of the coverage profiles from a qsea object for exploratory data analysis.

**Usage**

```
getPCA(qs, chr= getChrNames(qs),ROIs, minRowSum=20, keep ,  
norm_method=normMethod(logRPM =  
c("log", "library_size", "cnv", "preference", "psC10")), topVar=1000,  
samples=getSampleNames(qs), minEnrichment = 0)
```

**Arguments**

|               |  |
|---------------|--|
| qs            | DIPSset (mandatory)  |
| chr           | chromosomes to consider  |
| ROIs          | If specified, only windows overlapping ROIs are considered.  |
| minRowSum     | minimal number of total read counts per window over all samples  |
| keep          | windows to consider  |
| norm_method   | name of predefined normalization (e.g. "beta"), or user defined normalization by calling normMethod() function   |
| topVar        | only the top variable windows are considered   |
| samples       | names of samples to be considered  |
| minEnrichment | for transformation to absolute methylation level, you can specify the minimal number of expected reads for a fully methylated window. This avoids inaccurate estimates, due to low enrichment. |

**Details**

The principle component analysis is calculated using the singular value decomposition (svd).

**Value**

getPCA() returns a list object, containing the svd and information on the selected windows.

**Author(s)**

Mathias Lienhard

**See Also**

plotPCA

**Examples**

```
qs=getExampleQseaSet( repl=5)
pca=getPCA(qs, norm_method="beta")
colors=c(rep("red", 5), rep("green", 5))
plotPCA(pca, bgColor=colors)
#plotPCAFactors is more interesting, if ROIs have been specified in getPCA
plotPCAFactors(pca)
```

---

isSignificant

*Finds Significant Regions*

---

**Description**

This function looks for regions, where the test statistic is below the defined thresholds

**Usage**

```
isSignificant(glm, contrast = NULL, fdr_th = NULL, pval_th = NULL,
              absLogFC_th = NULL, direction = "both")
```



**Arguments**

|             |   |
|-------------|---|
| glm         | A qseaGLM object (mandatory)                          |
| contrast    | name of contrast to be used                           |
| fdr_th      | a threshold for the false discovery rate              |
| pval_th     | a p value threshold                                   |
| absLogFC_th | the threshold for the absolute value of logFC         |
| direction   | direction of change: either "both", "loss", or "gain" |

**Details**

If a threshold is NULL, it is ignored.

For the direction parameter, the following synonyms are valid:

"loss" == "less" == "hypo"

"gain" == "more" == "hyper"

**Value**

A vector with indices of significant windows, which can be passed to keep parameter of makeTable() function

**Author(s)**

Mathias Lienhard

**See Also**

makeTable

**Examples**

```
qs=getExampleQseaSet()
design=model.matrix(~group, getSampleTable(qs))
TvN_glm=fitNBglm(qs, design, norm_method="beta")
TvN_glm=addContrast(qs,TvN_glm, coef=2, name="TvN")
sig=isSignificant(TvN_glm, fdr_th=0.01)
```

---

makeTable

*Create a Results Table*

---

**Description**

This function creates a table from the qsea objects qseaSet and qseaTvN\_glm

**Usage**

```
makeTable(qs,glm,norm_methods="counts",samples,groupMeans, keep, ROIs,
annotation, minPvalSummarize, CNV=FALSE, verbose=TRUE, minEnrichment=3,
chunksize=1e5)
```

**Arguments**

|                  |  |
|------------------|--|
| qs               | a qseaSet object (mandatory)   |
| glm              | a list of one or more qseaGLM objects (optional)   |
| norm_methods     | either a character vector of pre-defined normalization combinations, or a list defining normalization combinations. This affects both individual and mean values.                              |
| samples          | The indices of the samples for which individual values are to be written out in the specified order  |
| groupMeans       | a named list of indices vectors, defining groups for which mean values are to be written out   |
| keep             | a vector of indices of the windows that are considered (as created by isSignificant)   |
| ROIs             | A GRanges object, containing regions of interest (ROIs). Only windows overlapping ROIs are considered.   |
| annotation       | a named list of GRRange objects, containing annotations (e.g. genes, CpG islands, ...) that are added to the table.  |
| minPvalSummarize | If ROIs are given, you can specify a QseaTvN_glm object. For each ROI the window with the most significant differential coverage is written out  |
| CNV              | If set TRUE, the CNV logFC for the samples specified by samples are written out.   |
| verbose          | verbosity level  |
| minEnrichment    | for transformation to absolute methylation level, you can specify the minimal number of expected reads for a fully methylated window. This avoids inaccurate estimates, due to low enrichment. |
| chunksize        | For efficient memory usage, the table is built up in chunks. With this parameter, the maximum number of windows processed in one chunk is specified.   |

**Details**

Note that, if overlapping ROIs are specified, windows might emerge in the table several times.

**Value**

A result table containing the specified normalized values for the selected windows and samples/groups

**Author(s)**

Mathias Lienhard

**See Also**

isSignificant

**Examples**

```

#create example set
qs=getExampleQseaSet()
design=model.matrix(~group, getSampleTable(qs))
TvN_glm=fitNBglm(qs, design, norm_method="beta")
TvN_glm=addContrast(qs,TvN_glm, coef=2, name="TvN")
sig=isSignificant(TvN_glm, fdr_th=0.01)

##Table containing all significant windows
tab1=makeTable(qs=qs, glm=TvN_glm,
  keep=sig, samples=getSampleNames(qs))
##additional CNV logFC for the selected samples
tab2=makeTable(qs=qs, glm=TvN_glm,
  keep=sig, samples=getSampleNames(qs), CNV=TRUE)
##explicit selection of normalization:
##counts (i.e. no normalization, only counts)
tab3=makeTable(qs=qs, glm=TvN_glm, keep=sig,
  samples=getSampleNames(qs), norm_method="counts")

##counts AND %methylation values for individual samples and group means
tab4=makeTable(qs=qs, glm=TvN_glm, keep=sig,
  samples=getSampleNames(qs), groupMeans=getSampleGroups(qs),
  norm_method=c("counts", "beta"))

```

normMethod

*Definition of normalization procedure***Description**

This function allows to define normalization methods by specifying components.

**Usage**

```
normMethod(methods, ...)
```

**Arguments**

|         |  |
|---------|--|
| methods | names of predefined normalization methods ( for a list of predefined methods, see details)   |
| ...     | sets of normalization components, that can be combined to user defined normalization methods |

**Details**

Predefined normalization methods:

“counts”: no normalization, simply raw count values

“reads”: same as counts

“rpm”: reads per million mappable reads

“nrpm”: CNV normalized reads per million mappable reads

“beta”: transformation to % methylation, posterior mean point estimator

“logitbeta”: logit transformed beta values

“betaLB”: 2.5 lower bound for the point estimator

“betaUB”: 97.5 upper bound for the point estimator

Allowd components for user defined normalization methods:

“library\_size”: scale by effective library size

“region\_length”: scale by window size

“preference”: scale by positional sequencing preference

“cnv”: scale by CNV ratio

“enrichment”: use enrichment profiles for transformation to absolute methylation level

“qXY”: quantile estimator for transformation to absolute methylation level. XY must be replaced by the quantile (see example with self defined lower and upper bound)

“offset”: consider background reads

WARNING: not all combinations are allowed (eg qXY requires enrichment) and not all allowed combinations are meaningful. Inexperienced users should stick to predefined normalization methods.

## Value

a list object, containing the components for the specified normalization procedure

## Author(s)

Mathias Lienhard

## See Also

makeTable

## Examples

```
#simply raw counts
nm=normMethod("counts")
#beta-values (% methylation) including lower and upper bounds
nm=normMethod(c("beta", "betaLB", "betaUB"))
#self defined lower and upper bound: 10% and 90% quantile
nm=normMethod("beta",
betaLB_10=c("enrichment", "cnv", "library_size",
"region_length", "preference", "q10", "offset"),
betaUB_90=c("enrichment", "cnv", "library_size",
"region_length", "preference", "q90", "offset")
)
```

**Description**

This function plots the Copy Number Variations (CNVs) of the samples in a heatmap like representation. Amplified regions are depicted in red, whereas deletions are depicted green, and CNV free regions blue. The samples are ordered by an hierarchical clustering.

**Usage**

```
plotCNV(qs, dist = c("euclid", "cor")[1], clust_method = "complete",  
        chr = getChrNames(qs), samples =getSampleNames(qs),  
        cex = 1, labels = c(TRUE, TRUE,TRUE, TRUE), naColor = "darkgrey",  
        indicateLogFC = TRUE )
```

**Arguments**

|               |   |
|---------------|---|
| qs            | a qseaSet object (mandatory)  |
| dist          | distance measure for clustering. dQuoteeuclidian or dQuotecorrelation based (1-cor)                     |
| clust_method  | method to be passed to hclust   |
| chr           | vector of chromosomes to be depicted  |
| samples       | samples for which CNVs are depicted   |
| cex           | font size of labels   |
| labels        | Boolean vector of length four (bottom, left, top, right), specifying the sides of the map to be labeled |
| naColor       | Color for regions without CNV information   |
| indicateLogFC | indicate the CNV logFC values in the legend   |

**Value**

This function returns the pairwise distances of the CNV profiles, on which the clustering is based on.

**Author(s)**

Mathias Lienhard

**Examples**

```
qs=getExampleQseaSet()  
plotCNV(qs, labels=c(FALSE, TRUE, TRUE, FALSE))
```

---

plotCoverage

*Plots a genome-browser-like image of a region*


---

### Description

This function plots the normalized coverage of specified samples in a specified region, together with annotations, in a genome-browser-like fashion

### Usage

```
plotCoverage(qs,test_results, chr, start, end, samples,samples2,
norm_method="nrpkm", yoffset, xlab="Position",
ylab="MeDIP seq", col="black", main, reorder="non", indicate_reorder=TRUE,
distfun=dist, clustmethod="complete", scale=TRUE, steps=TRUE, space=0.05,
baselines=TRUE, scale_val, scale_unit=NULL, logFC_pc=.1, cex=1, smooth_width,
smooth_function=mean, regions, regions_lwd=1, regions_col,
regions_offset, regions_names, regions_dash=0.1)
```

### Arguments

|                  |  |
|------------------|--|
| qs               | a qseaSet object   |
| chr              | the chromosome of the region to be depicted  |
| start            | the start position of the region to be depicted  |
| end              | the end position of the region to be depicted  |
| samples          | the indices of the samples to be depicted  |
| samples2         | if specified, used to calculated logFC (samples/samples2) profiles, must be of same length as samples  |
| logFC_pc         | if samples2 is specified and logFC are calculated, this parameter specifies the pseudocount to avoid division by zero  |
| norm_method      | a vector of normalization methods to be combined   |
| yoffset          | horizontal offset, used to adjust the space between the profiles   |
| xlab             | title for the x axis   |
| ylab             | title for the y axis   |
| main             | an overall title for the plot  |
| col              | color vector for the samples (is recycled)   |
| reorder          | indicate whether, and if yes how, the samples are reordered. Valid values are "non", "clust", "max", "minP", or a genomic position within the range that is depicted |
| test_results     | a qseaGLM object, used to find the region with minimal p value (only if reorder="minP")  |
| indicate_reorder | indicate the window that has been used for reordering by an arrow.   |
| distfun          | if reorder="clust": for hierarchical clustering for reordering   |
| clustmethod      | if reorder="clust": for hierarchical clustering for reordering   |
| scale            | if set TRUE, print a bar scale   |

|                 |  |
|-----------------|--|
| scale_val       | length of the bar scale  |
| scale_unit      | unit of the bar scale  |
| steps           | plot the coverage as step function (steps=TRUE), or as lines   |
| space           | fraction of the plot set aside for sample names etc.   |
| baselines       | depict the baselines (zero) of the coverage profiles   |
| cex             | font size  |
| smooth_width    | number of windows to be considered for sliding window smoothing  |
| smooth_function | function to be applied on the sliding windows for smoothing  |
| regions         | named list of GenomicRanges objects, containing annotation (eg exons) to be depicted below the coverage profiles |
| regions_lwd     | vector of line width for the   |
| regions_col     | vector of colors for the regions   |
| regions_offset  | offset value, defining the space between the regions   |
| regions_names   | vector of column names, that store the names of the regions  |
| regions_dash    | vector, specifying the length of the end dashes of the regions   |

**Value**

list containing a table containing the plotted coverage values, the position that has been used for ordering, and the image coordinates

**Author(s)**

Mathias Lienhard

**Examples**

```
qs=getExampleQseaSet(repl=5)
colors=c(rep("red", 5), rep("green", 5))
plot(1)
plotCoverage(qs,samples=getSampleNames(qs),
  chr="chr1", start=1960001, end=1970001,col=colors,
  norm_method="beta", yoffset=1,space=.2, reorder=1964500)
plotCoverage(qs,samples=getSampleNames(qs),
  chr="chr1", start=1960001, end=1970001,col=colors,
  norm_method="beta", yoffset=1,space=.2, reorder="clust")
```

---

plotEnrichmentProfile *Plotting functions for enrichment profiles*

---

**Description**

Plots the estimated sequence pattern dependent enrichment profile for one or several samples as a matrix of plots

**Usage**

```
plotEnrichmentProfile(qs, sample, sPoints=seq(0,30,1),
  fitPar=list(lty=2, col="green"), cfPar=list(lty=1), densityPar, meanPar, ... )
plotEPmatrix(qs, sa=getSampleNames(qs), nrow=ceiling(sqrt(length(sa))),
  ncol=ceiling(length(sa)/nrow), ...)
```

**Arguments**

|            |   |
|------------|---|
| qs         | The qseaSet object  |
| sample     | The index of the sample for which the enrichment profile should be depicted                             |
| sPoints    | The values at which the enrichment profile function is evaluated  |
| fitPar     | List of parameters for depiction of the fitted enrichment profile function (see details)                |
| cfPar      | List of parameters for depiction of the empirical enrichment profile (see details)                      |
| densityPar | List of parameters for depiction high density scatterplot of coverage and pattern density (see details) |
| meanPar    | List of parameters for depiction of the mean coverage per pattern density bin (see details)             |
| sa         | vector of samples to be depicted in matrix plot   |
| nrow       | number of rows in matrix plot   |
| ncol       | number of columns in matrix plot  |
| ...        | Further graphical parameters may also be supplied   |

**Details**

Parameter lists for lines in the plot (e.g. fitPar, cfPar and meanPar) are passed to graphics::lines(), densityPar are passed to graphics::smoothScatter() function.

**Value**

plotEnrichmentProfile returns the coordinates of the enrichment profile. plotEPmatrix returns enrichment profile coordinates for all depicted samples.

**Author(s)**

Mathias Lienhard

**See Also**

addEnrichmentParameters

**Examples**

```
#create example object with different sequencing depth
qs=getExampleQseaSet(expSamplingDepth=50*10^(1:4), repl=4)
#enrichment profile for one sample
plotEnrichmentProfile(qs, "Sim4T")
#enrichment profile for all samples
plotEPmatrix(qs)
```



**Description**

The principle components can be depicted using the plotting methods `plotPCA` and `plotPCAfactors`

**Usage**

```
## S4 method for signature 'qseaPCA'
plotPCA(object,plotComponents=c(1,2), fgColor="black",
        bgColor = "white", legend, plotLabels=TRUE, radius=5, labelOffset=.5,
        labelPos=1, labelAdj, labelColor="black", cex=1, ...)

## S4 method for signature 'qseaPCA'
plotPCAfactors(object,plotComponents=c(1,2),
               fgColor="black",bgColor = "white", plotTopLabels=100, labelsOfInterest,
               radius=1, labelOffset=.5,labelPos=1,labelColor="black", cex=1, ...)
```

**Arguments**

|                               |   |
|-------------------------------|---|
| <code>object</code>           | the <code>qseaPCA</code> object, resulting from the <code>getPCA</code> function                |
| <code>plotComponents</code>   | vector of the two components of the PCA   |
| <code>fgColor</code>          | vector of foreground colors for the circles   |
| <code>bgColor</code>          | vector of background colors for the circles   |
| <code>legend</code>           | add a legend to the plot  |
| <code>plotLabels</code>       | if set <code>TRUE</code> , the labels of the samples are written in the plot                    |
| <code>radius</code>           | defines the size of the plotted circles   |
| <code>labelOffset</code>      | defines the offset of the labels to the circles   |
| <code>labelPos</code>         | specify position of the labels in the plot (see <code>graphics::text</code> )                   |
| <code>labelAdj</code>         | alternative way to specify position of the labls in the plot (see <code>graphics::text</code> ) |
| <code>labelColor</code>       | a vector of colors for the labels   |
| <code>cex</code>              | font size of the labels   |
| <code>plotTopLabels</code>    | labels of factors with strongest contribution to plotted components are shown                   |
| <code>labelsOfInterest</code> | vector of factor names that are highlighted and labeled in the plot                             |
| <code>...</code>              | further graphical parameters  |

**Value**

The functions return a list with the coordinates of the depicted components

**Author(s)**

Mathias Lienhard

**See Also**

plotPCA

**Examples**

```
qs=getExampleQseaSet( repl=5)
pca=getPCA(qs, norm_method="beta")
colors=c(rep("red", 5), rep("green", 5))

plotPCA(pca, bgColor=colors)
#plotPCAfactors is more interesting, if ROIs have been specified in getPCA
plotPCAfactors(pca)
```

---

qseaGLM-class

*qseaGLM class and its methods*

---

**Description**

The qseaGLM class is used in qsea to store fitted coefficients of the GLM.

**Slots**

fullModelDesign: design matrix of full model  
 fullModel : list containing parameters and fitted coefficients of full model  
 parameters: list of parameters used to create the object  
 contrast: list of lists containing parameters and the fitted model coefficients of the reduced models  
 windows: vector of window indices, for which GLMs have been fitted

**Author(s)**

Matthias Lienhard

**Examples**

```
showClass("qseaGLM")
```

---

qseaPCA-class

*qseaPCA class and its methods*

---

**Description**

The qseaPCA class is used in qsea to store results of the principle component analysis.

**Slots**

svd: singular value decomposition  
 sample\_names : names of the samples  
 factor\_names: names of the genomic windows involved

**Author(s)**

Matthias Lienhard

**Examples**

```
showClass("qseaPCA")
```

---

qseaSet-class

*qseaSet class and its methods*


---

**Description**

The qseaSet class is used in qsea to store information about the coverage, the dependent organism, the chromosomes included in the input file, the length of the included chromosomes (automatically loaded), the number of regions, and optionally CNV information.

**Slots**

**sampleTable:** Object of class "data.frame": the sample table

**count\_matrix:** Object of class "matrix": matrix containing the coverage for all samples

**zygosity:** Object of class "matrix": matrix containing the zygosity for all chromosomes and all samples

**regions:** Object of class "GenomicRanges": the genomic regions for the coverage matrix

**parameters:** Object of class "list": the parameter list used to create this object

**cnv:** Object of class "GenomicRanges": CNV ranges and logFCs

**enrichment :** Object of class "list": parameters of the sequence pattern enrichment analysis

**libraries :** Object of class "matrix": parameters of the sequencing libraries

**Methods**

**getSampleTable** signature(object = "qseaSet"): extracts the sample table of a qsea set

**getSampleNames** signature(object = "qseaSet"): extracts the sample names of a qsea set

**getSampleGroups** signature(object = "qseaSet"): extracts the sample groups of a qsea set

**getChrNames** signature(object = "qseaSet"): returns the analysed chromosomes

**getCounts** signature(object = "qseaSet"): extracts the count matrix a qsea set

**getRegions** signature(object = "qseaSet"): extracts the regions object of a qsea set

**getParameters** signature(object = "qseaSet"): extracts the parameter list of a qsea set

**getLibSize** signature(object = "qseaSet"): extracts the library size (eg the total number of read counts per sample)

**getNormFactors** signature(object = "qseaSet"): extracts the list with the different normalization factors

**hasCNV** signature(object = "qseaSet"): TRUE if CNV information is present, FALSE otherwise

**getCNV** signature(object = "qseaSet"): extracts the CNV regions and logFCs

**getOffset** signature(object = "qseaSet"): extracts offset of rpkm scaled background reads

**getWindowSize** signature(object = "qseaSet"): returns the window size of the object

**getZyosity** signature(object = "qseaSet"): returns the zygosity matrix of the object

**setZyosity** signature(object = "qseaSet", zygosityMatrix): sets the zygosity matrix, and resets CNV

**Author(s)**

Matthias Lienhard

**Examples**

```
showClass("qseaSet")
```

---

 regionStats

*Counts the Windows in Regions of Interest*


---

**Description**

This function takes a list of window indices and a list of ROIs and counts the number of overlapping windows

**Usage**

```
regionStats(qs, subsets = list(covered = which(rowSums(getCounts(qs)) >= 20)),
  ROIs = list(), minoverlap = 0, maxgap = -1)
```

**Arguments**

|            |                               |
|------------|-------------------------------|
| qs         | A qsea Set object             |
| subsets    | A list of window indices      |
| ROIs       | A list of Regions of Interest |
| minoverlap | Passed to findOverlaps        |
| maxgap     | Passed to findOverlaps        |

**Value**

a matrix, containing the total number of windows overlapping the ROIs and the numbers of windows from the subset list overlapping ROIs

**Author(s)**

Mathias Lienhard

**See Also**

findOverlaps

**Examples**

```
qs=getExampleQseaSet()
#as an example, we analyze the fraction of reads covered by at least 10
#or at least 20 reads, for bins of CpG density
ROIs=list()
regs=getRegions(qs)
cpg=getRegions(qs)$CpG_density
bins=seq(0,30,5)
for(i in 1:(length(bins)-1)){
```

```
n=paste0(bins[i],"-",bins[i+1]," CpGs")
ROIs[[n]]=regs[which(cpg>=bins[i] & cpg < bins[i+1])]
}
subsets = list(
  ">10" = which(rowSums(getCounts(qs)) >= 10),
  ">20" = which(rowSums(getCounts(qs)) >= 20))
coverage_stats=regionStats(qs, subsets, ROIs)
coverage_stats_rel=coverage_stats[,-1]/coverage_stats[,1]
x=barplot(t(coverage_stats_rel)*100,ylab="fraction of windows[%]",
  beside=TRUE, legend=TRUE, las=2, args.legend=list(x="topleft"),
  main="Covered Windows")
```

# Index

- \* **classes**
  - qseaGLM-class, 26
  - qseaPCA-class, 26
  - qseaSet-class, 27
- \* **package**
  - qsea-package, 2
- addCNV, 3
- addContrast, 4
- addCoverage, 5
- addEnrichmentParameters, 6
- addLibraryFactors, 7
- addNewSamples, 8
- addOffset, 9
- addParameters, qseaGLM-method (qseaGLM-class), 26
- addParameters, qseaSet-method (qseaSet-class), 27
- addPatternDensity, 10
- addSeqPref, 11
- createQseaSet, 12
- fitNBglm, 13
- getChrNames (qseaSet-class), 27
- getChrNames, qseaSet-method (qseaSet-class), 27
- getCNV (qseaSet-class), 27
- getCNV, qseaSet-method (qseaSet-class), 27
- getCounts (qseaSet-class), 27
- getCounts, qseaSet-method (qseaSet-class), 27
- getExampleQseaSet, 14
- getLibSize (qseaSet-class), 27
- getLibSize, qseaSet-method (qseaSet-class), 27
- getNormFactors (qseaSet-class), 27
- getNormFactors, qseaSet-method (qseaSet-class), 27
- getOffset (qseaSet-class), 27
- getOffset, qseaSet-method (qseaSet-class), 27
- getParameters (qseaSet-class), 27
- getParameters, qseaGLM-method (qseaGLM-class), 26
- getParameters, qseaSet-method (qseaSet-class), 27
- getPCA, 15
- getRegions (qseaSet-class), 27
- getRegions, qseaSet-method (qseaSet-class), 27
- getSampleGroups (qseaSet-class), 27
- getSampleGroups, qseaSet-method (qseaSet-class), 27
- getSampleNames (qseaSet-class), 27
- getSampleNames, qseaGLM-method (qseaGLM-class), 26
- getSampleNames, qseaPCA-method (qseaPCA-class), 26
- getSampleNames, qseaSet-method (qseaSet-class), 27
- getSampleTable (qseaSet-class), 27
- getSampleTable, qseaSet-method (qseaSet-class), 27
- getWindowWidth (qseaSet-class), 27
- getWindowWidth, qseaSet-method (qseaSet-class), 27
- getZygoty (qseaSet-class), 27
- getZygoty, qseaSet-method (qseaSet-class), 27
- hasCNV (qseaSet-class), 27
- hasCNV, qseaSet-method (qseaSet-class), 27
- isSignificant, 16
- makeTable, 17
- normMethod, 19
- plotCNV, 21
- plotCoverage, 22
- plotEnrichmentProfile, 23
- plotEPmatrix (plotEnrichmentProfile), 23
- plotPCA, 25
- plotPCA, qseaPCA-method (plotPCA), 25

plotPCAFactors (plotPCA), [25](#)  
plotPCAFactors, qseaPCA-method  
    (plotPCA), [25](#)

QSEA (qsea-package), [2](#)  
qsea (qsea-package), [2](#)  
qsea-package, [2](#)  
qseaGLM (qseaGLM-class), [26](#)  
qseaGLM-class, [26](#)  
qseaPCA (qseaPCA-class), [26](#)  
qseaPCA-class, [26](#)  
qseaSet (qseaSet-class), [27](#)  
qseaSet-class, [27](#)

regionStats, [28](#)

setZygoty (qseaSet-class), [27](#)  
setZygoty, qseaSet-method  
    (qseaSet-class), [27](#)  
show, qseaGLM-method (qseaGLM-class), [26](#)  
show, qseaPCA-method (qseaPCA-class), [26](#)  
show, qseaSet-method (qseaSet-class), [27](#)