

# Package ‘epiSeeker’

May 1, 2026

**Type** Package

**Title** epiSeeker: an R package for Annotation, Comparison and Visualization of multi-omics epigenetic data

**Version** 1.1.0

**Description** This package implements functions to analyze multi-omics epigenetic data. Data of fragment type and base type are supported by epiSeeker. It provides functions to retrieve the nearest genes around the peak, annotate genomic region of the peak, statistical methods to estimate the significance of overlap among peak data sets, and motif analysis. It incorporates the GEO database for users to compare their own dataset with those deposited in the database. The comparison can be used to infer cooperative regulation and thus can be used to generate hypotheses. Several visualization functions are implemented to summarize the coverage of the peak experiment, average profile and heatmap of peaks binding to TSS regions, genomic annotation, distance to TSS, overlap of peaks or genes, and the single-base resolution epigenetic data by considering the strand, motif, and additional information.

**Depends** R (>= 4.6.0)

**Imports** AnnotationDbi, aplot, bsseq, BiocGenerics, Biostrings, boot, dplyr, enrichplot, IRanges, GenomeInfoDb, GenomicRanges, GenomicFeatures, ggplot2, graphics, grDevices, magrittr, methods, plotrix, parallel, RColorBrewer, rlang, RSQLite, rtracklayer, S4Vectors, scales, stats, SummarizedExperiment, tibble, tidyselect, tidyr, utils, yulab.utils (>= 0.2.0), grid

**Suggests** ape, BSgenome, BSgenome.Hsapiens.UCSC.hg38, clusterProfiler, data.table, GEOMETadb, GEOquery, gggenes, ggimage, ggiraph, ggplotify, ggtree, gginnards, gridBase, gtools, ggupset, ggVennDiagram, JASPAR2024, knitr, org.Hs.eg.db, prettydoc, ReactomePA, rmarkdown, testthat, TFBSTools, TxDb.Hsapiens.UCSC.hg38.knownGene, universalmotif

**URL** <https://github.com/YuLab-SMU/epiSeeker>

**BugReports** <https://github.com/YuLab-SMU/epiSeeker/issues>

**Encoding** UTF-8

**VignetteBuilder** knitr

**ByteCompile** true  
**License** Artistic-2.0  
**biocViews** Annotation, ChIPSeq, Software, Visualization,  
 MultipleComparison, Coverage, MotifAnnotation, GeneRegulation  
**RoxygenNote** 7.3.3  
**LazyData** false  
**git\_url** <https://git.bioconductor.org/packages/epiSeeker>  
**git\_branch** devel  
**git\_last\_commit** 03bc0c5  
**git\_last\_commit\_date** 2026-04-28  
**Repository** Bioconductor 3.24  
**Date/Publication** 2026-05-01  
**Author** Guangchuang Yu [aut, cre, fnd] (ORCID:  
<https://orcid.org/0000-0002-6485-8781>),  
 Ming Li [ctb],  
 Qianwen Wang [ctb],  
 Yun Yan [ctb],  
 Hervé Pagès [ctb],  
 Michael Kluge [ctb],  
 Thomas Schwarzl [ctb],  
 Zhougeng Xu [ctb],  
 Chun-Hui Gao [ctb]  
**Maintainer** Guangchuang Yu <guangchuangyu@gmail.com>

## Contents

epiSeeker-package . . . . .	4
. . . . .	5
.epiSeekerEnv . . . . .	5
annotateSeq . . . . .	6
arrange.GRanges . . . . .	8
as.data.frame.csAnno . . . . .	8
as.GRanges . . . . .	9
bin_vector . . . . .	9
bmData . . . . .	10
bmData-class . . . . .	11
check_bin . . . . .	11
check_extension . . . . .	12
check_windows . . . . .	12
combine_csAnno . . . . .	13
create_regex_patterns_negative . . . . .	13
create_regex_patterns_positive . . . . .	14
csAnno-class . . . . .	14
demo_bmdata . . . . .	15
demo_peak . . . . .	16
downloadGEObedFiles . . . . .	16
downloadGSMbedFiles . . . . .	17
dropAnno . . . . .	17

enrichAnnoOverlap . . . . .	18
enrichPeakOverlap . . . . .	19
epiSeekerCache . . . . .	20
extend_gr . . . . .	20
filter.GRanges . . . . .	21
getAnnoStat . . . . .	21
getBioRegion . . . . .	22
getBmMatrix . . . . .	23
getBmMatrix.bmData . . . . .	24
getBmMatrix.BSseq . . . . .	25
getGeneAnno . . . . .	26
getGenomicAnnotation . . . . .	26
getGEOgenomeVersion . . . . .	27
getGEOInfo . . . . .	28
getGEOspecies . . . . .	28
getMotifMatrix . . . . .	29
getNearestFeatureIndicesAndDistances . . . . .	29
getPromoters . . . . .	30
getSampleFiles . . . . .	31
getTagMatrix . . . . .	31
getTagMatrix.internal . . . . .	33
getTagMatrix_body . . . . .	34
getTagMatrix_body_internal . . . . .	35
getTagMatrix_site . . . . .	35
grange2mt . . . . .	36
gsminfo . . . . .	36
loadTxDb . . . . .	38
makeBmDataFromData . . . . .	38
makeBmDataFromData.internal . . . . .	39
makeBmDataFromFiles . . . . .	40
mutate.GRanges . . . . .	41
overlap . . . . .	42
parse_peak . . . . .	42
peakAnno . . . . .	43
peakAnnoList . . . . .	44
plotAnnoBar . . . . .	44
plotAnnoBar.data.frame . . . . .	45
plotAnnoPie . . . . .	46
plotAnnoPie.csAnno . . . . .	47
plotBmProf . . . . .	48
plotCov . . . . .	50
plotDistToTSS . . . . .	53
plotDistToTSS.data.frame . . . . .	54
plotGeneTrack . . . . .	55
plotMotifProf . . . . .	56
plotPeakHeatmap . . . . .	57
plotPeakHeatmap_sub . . . . .	58
plotPeakHeatmap_sub.internal . . . . .	59
plotPeakProf . . . . .	60
pwm_obj . . . . .	61
readPeakFile . . . . .	61
reexports . . . . .	62

rename.GRanges . . . . .	62
seq2gene . . . . .	63
seq2gene_result . . . . .	64
show . . . . .	64
shuffle . . . . .	65
tagMatrix . . . . .	66
upsetplot . . . . .	66
vennpie . . . . .	67
vennplot . . . . .	68
vennplot.peakfile . . . . .	68

<b>Index</b>	<b>70</b>
--------------	-----------

---

epiSeeker-package	<i>epiSeeker: epiSeeker: an R package for Annotation, Comparison and Visualization of multi-omics epigenetic data</i>
-------------------	---

---

## Description

This package implements functions to analyze multi-omics epigenetic data. Data of fragment type and base type are supported by epiSeeker. It provides functions to retrieve the nearest genes around the peak, annotate genomic region of the peak, statistical methods to estimate the significance of overlap among peak data sets, and motif analysis. It incorporates the GEO database for users to compare their own dataset with those deposited in the database. The comparison can be used to infer cooperative regulation and thus can be used to generate hypotheses. Several visualization functions are implemented to summarize the coverage of the peak experiment, average profile and heatmap of peaks binding to TSS regions, genomic annotation, distance to TSS, overlap of peaks or genes, and the single-base resolution epigenetic data by considering the strand, motif, and additional information.

## Author(s)

**Maintainer:** Guangchuang Yu <guangchuangyu@gmail.com> ([ORCID](#)) [funder]

Other contributors:

- Ming Li <limiang929@gmail.com> [contributor]
- Qianwen Wang <treywea@gmail.com> [contributor]
- Yun Yan <youryanyun@gmail.com> [contributor]
- Hervé Pagès <hpages.on.github@gmail.com> [contributor]
- Michael Kluge <michael.kluge@bio.ifi.lmu.de> [contributor]
- Thomas Schwarzl <schwarzl@embl.de> [contributor]
- Zhougeng Xu <xuzhougeng@163.com> [contributor]
- Chun-Hui Gao <gaospecial@gmail.com> [contributor]

## See Also

Useful links:

- <https://github.com/YuLab-SMU/epiSeeker>
- Report bugs at <https://github.com/YuLab-SMU/epiSeeker/issues>

---

**Description**

capture name of variable

**Usage**

```
.(..., .env = parent.frame())
```

**Arguments**

...	expression
.env	environment

**Value**

expression

**Examples**

```
x <- 1
eval(. (x)[[1]])
```

---

.epiSeekerEnv	<i>Env function for epiSeeker</i>
---------------	-----------------------------------

---

**Description**

Env function for epiSeeker

**Usage**

```
.epiSeekerEnv(TxDB, item = "epiSeekerEnv", force = FALSE)
```

**Arguments**

TxDB	TxDB object
item	item name
force	force to update txdb item in cache or not.

**Value**

Returns ‘invisible(NULL)’ invisibly. The primary purpose of this function is to manage the TXDB cache through side effects (creating, updating, or removing cached objects), rather than returning a value.

---

annotateSeq                      *annotateSeq*

---

## Description

Annotate peaks

## Usage

```
annotateSeq(
  peak,
  tssRegion = c(-3000, 3000),
  TxDb = NULL,
  level = "transcript",
  assignGenomicAnnotation = TRUE,
  genomicAnnotationPriority = c("Promoter", "5UTR", "3UTR", "Exon", "Intron",
    "Downstream", "Intergenic"),
  annoDb = NULL,
  addFlankGeneInfo = FALSE,
  flankDistance = 5000,
  sameStrand = FALSE,
  ignoreOverlap = FALSE,
  ignoreUpstream = FALSE,
  ignoreDownstream = FALSE,
  overlap = "TSS",
  verbose = TRUE,
  columns = c("ENTREZID", "ENSEMBL", "SYMBOL", "GENENAME")
)
```

## Arguments

peak	peak file or GRanges object
tssRegion	Region Range of TSS
TxDb	TxDb or EnsDb annotation object
level	one of transcript and gene
assignGenomicAnnotation	logical, assign peak genomic annotation or not
genomicAnnotationPriority	genomic annotation priority
annoDb	annotation package
addFlankGeneInfo	logical, add flanking gene information from the peaks
flankDistance	distance of flanking sequence
sameStrand	logical, whether find nearest/overlap gene in the same strand
ignoreOverlap	logical, whether ignore overlap of TSS with peak
ignoreUpstream	logical, if True only annotate gene at the 3' of the peak.
ignoreDownstream	logical, if True only annotate gene at the 5' of the peak.

overlap	one of 'TSS' or 'all', if overlap="all", then gene overlap with peak will be reported as nearest gene, no matter the overlap is at TSS region or not.
verbose	print message or not
columns	names of columns to be obtained from database

### Value

data.frame or GRanges object with columns of:

all columns provided by input.

annotation: genomic feature of the peak, for instance if the peak is located in 5'UTR, it will annotated by 5'UTR. Possible annotation is Promoter-TSS, Exon, 5' UTR, 3' UTR, Intron, and Inter-genic.

geneChr: Chromosome of the nearest gene

geneStart: gene start

geneEnd: gene end

geneLength: gene length

geneStrand: gene strand

geneId: entrezgene ID

distanceToTSS: distance from peak to gene TSS

if annoDb is provided, extra column will be included:

ENSEMBL: ensembl ID of the nearest gene

SYMBOL: gene symbol

GENENAME: full gene name

### Author(s)

G Yu

### See Also

[plotAnnoBar()] [plotAnnoPie()] [plotDistToTSS()]

### Examples

```
data(peakAnno)
peakAnno
```

---

arrange.GRanges      *Arrange GRanges object*

---

### Description

Arrange GRanges object

### Usage

```
## S3 method for class 'GRanges'
arrange(.data, ..., .by_group = FALSE)
```

### Arguments

.data	granges object
...	additional parameters
.by_group	If TRUE, will sort first by grouping variable. Applies to grouped data frames only.

### Value

grange object

### Examples

```
peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
peak <- readPeakFile(peakfile)
dplyr::arrange(peak, seqnames)
```

---

as.data.frame.csAnno      *as.data.frame.csAnno*

---

### Description

convert csAnno object to data.frame

### Usage

```
## S3 method for class 'csAnno'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

### Arguments

x	csAnno object
row.names	row names
optional	should be omitted.
...	additional parameters

**Value**

data.frame

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

---

as.GRanges

*as.GRanges*

---

**Description**

convert csAnno object to GRanges

**Usage**

```
as.GRanges(x)
```

**Arguments**

x                   csAnno object

**Value**

GRanges object

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
data(peakAnno)
as.GRanges(peakAnno)
```

---

bin\_vector

*bin vector function*

---

**Description**

bin vector function

**Usage**

```
bin_vector(vec, nbin = 800)
```

**Arguments**

vec                   vector.  
nbin                   number of bin.

**Value**

bin list

---

`bmData`

---

*Constructor for bmData objects*

---

**Description**

This is constructor fo bmData objects.

**Usage**

```

bmData(
  value1 = NULL,
  value2 = NULL,
  pos = NULL,
  chr = NULL,
  gr = NULL,
  sampleNames = NULL,
  valueNames = NULL,
  ...
)

```

**Arguments**

<code>value1</code>	the first value to be stored, a matrix-like object
<code>value2</code>	the second value to be stored, a matrix-like object
<code>pos</code>	A vector of locations
<code>chr</code>	A vector of chromosomes
<code>gr</code>	An object of type [GenomicRanges::GRanges]
<code>sampleNames</code>	A vector of sample names
<code>valueNames</code>	the name of value1 or value2 or both. The order maps to the value.
<code>...</code>	other parameters from [bsseq::BSseq]

**Value**

bmData object

**Examples**

```
data(demo_bmdata)
```

---

bmData-class	<i>bmData Class</i>
--------------	---------------------

---

**Description**

This class added extra data to [bsseq::BSseq-class]. Change the assays by storing M/Cov to any value1/2

**Value**

bmData object

**See Also**

bmData class inherits [SummarizedExperiment::RangedSummarizedExperiment-class], other slots see [SummarizedExperiment::RangedSummarizedExperiment]

---

check_bin	<i>check bin parameter method</i>
-----------	-----------------------------------

---

**Description**

check bin parameter method

**Usage**

```
check_bin(nbin, windows, verbose)
```

**Arguments**

nbin	numbers of bin.
windows	a list of region in granges.
verbose	show details or not

**Value**

message or nothing

---

check_extension	<i>check upstream and downstream extension</i>
-----------------	--

---

**Description**

check upstream and downstream extension

**Usage**

```
check_extension(upstream, downstream, type)
```

**Arguments**

upstream	upstream extension. One of actual number or rel() object.
downstream	downstream extension. One of actual number or rel() object.
type	one of "start_site", "end_site", "body".

**Value**

message or null

---

check_windows	<i>check windows function</i>
---------------	-------------------------------

---

**Description**

check windows function

**Usage**

```
check_windows(windows)
```

**Arguments**

windows	windows
---------	---------

**Value**

message or null

---

combine_csAnno	<i>combine_csAnno</i>
----------------	-----------------------

---

**Description**

Combine csAnno Object

**Usage**

```
combine_csAnno(x, ...)
```

**Arguments**

x	csAnno object
...	csAnno objects

**Details**

<https://github.com/YuLab-SMU/ChIPseeker/issues/157>

**Value**

csAnno object

**Examples**

```
data(peakAnno)
combine_csAnno(peakAnno, peakAnno)
```

---

create_regex_patterns_negative	<i>create regex patterns in negative strand</i>
--------------------------------	---

---

**Description**

create regex patterns in negative strand

**Usage**

```
create_regex_patterns_negative(motif)
```

**Arguments**

motif	the motif(e.g C:CG/CH, A:GAGG/AGG) of the base modification
-------	---

**Value**

regex pattern

---

create\_regex\_patterns\_positive  
*create regex patterns in positive strand*

---

**Description**

create regex patterns in positive strand

**Usage**

create\_regex\_patterns\_positive(motif)

**Arguments**

motif                    the motif(e.g C:CG/CH, A:GAGG/AGG) of the base modification

**Value**

regex pattern

---

csAnno-class	<i>Class "csAnno" This class represents the output of epiSeeker Annotation</i>
--------------	--

---

**Description**

Class "csAnno" This class represents the output of epiSeeker Annotation

**Value**

annotation object

**Slots**

anno annotation  
 tssRegion TSS region  
 level transcript or gene  
 hasGenomicAnnotation logical  
 detailGenomicAnnotation Genomic Annotation in detail  
 annoStat annotation statistics  
 peakNum number of peaks

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**See Also**

[annotateSeq()]

---

demo_bmdata	<i>demo base modification data</i>
-------------	------------------------------------

---

### Description

A small example `bmData` object representing cytosine methylation measurements from Bisulfite-Seq data. This dataset is intended for demonstrating base-modification visualization, regional methylation profiling, and `epiSeeker` workflows operating on `bmData` objects. See `data-raw/example_data.R`

### Format

A `bmData` object containing one sample.

### Value

`bmData` object

### Provenance

The example dataset was constructed from publicly available Bisulfite-Seq data (GEO accession: GSM6940395, genome build: hg38). The raw methylation coverage file (`*.bismark.cov.gz`) was imported using `data.table::fread()`.

A small genomic window on chromosome 22 (`[10525991, 10526342]`) was selected to create a lightweight example dataset. The data were processed as follows:

1. Filter records where `chrom == 22` and positions fall within the chosen window.
2. Convert chromosome name to UCSC style (`"chr22"`).
3. Compute total coverage as: `Cov = methylated + unmethylated`.
4. Extract columns: chromosome, position, coverage, and methylation percentage.
5. Convert methylation percentage to a fraction.

### Data structure

A `bmData` S4 object containing one sample (`"acinar_methyl"`). Each entry stores:

`chr` Chromosome in UCSC format (e.g. `"chr22"`).

`pos` Genomic coordinate of the cytosine.

`Cov` Total read coverage at the site.

`Methylation` Methylation level as a fraction (0–1).

---

demo_peak	<i>demo peak file</i>
-----------	-----------------------

---

**Description**

Peak in Grange object. See data-raw/example\_data.R

**Format**

A GRanges object with 200 rows and 5 metadata columns.

**Value**

Grange object

**Provenance**

The demo peaks were extracted from GSM6418464 in the GEO database (<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM6418464>).

**Data structure**

A GRanges object with 220 genomic ranges and the following metadata columns:

seqnames chr name  
 ranges Peak ranges  
 strand strand information  
 mcol output from MACS2

---

downloadGEObedFiles	<i>downloadGEObedFiles</i>
---------------------	----------------------------

---

**Description**

Download all BED files of a particular genome version

**Usage**

```
downloadGEObedFiles(genome, destDir = getwd())
```

**Arguments**

genome	genome version
destDir	destination folder

**Value**

GEO files

**Author(s)**

G Yu

**Examples**

```
gse <- "GSE11431"
```

---

downloadGSMbedFiles     *downloadGSMbedFiles*

---

**Description**

Download BED supplementary files of a list of GSM accession numbers

**Usage**

```
downloadGSMbedFiles(GSM, destDir = getwd())
```

**Arguments**

GSM	GSM accession numbers
destDir	destination folder

**Value**

GEO data

**Author(s)**

G Yu

**Examples**

```
gsm <- "GSM288348"
```

---

dropAnno     *dropAnno*

---

**Description**

dropAnno

**Usage**

```
dropAnno(csAnno, distanceToTSS_cutoff = 10000)
```

**Arguments**

csAnno	output of annotateSeq
distanceToTSS_cutoff	distance to TSS cutoff

**Details**

drop annotation exceeding distanceToTSS\_cutoff

**Value**

csAnno object

**Author(s)**

Guangchuang Yu

**Examples**

```
data(peakAnno)
dropAnno(peakAnno)
```

---

```
enrichAnnoOverlap      enrichAnnoOverlap
```

---

**Description**

Calculate overlap significance of ChIP experiments based on their nearest gene annotation

**Usage**

```
enrichAnnoOverlap(
  queryPeak,
  targetPeak,
  TxDb = NULL,
  pAdjustMethod = "BH",
  chainFile = NULL,
  distanceToTSS_cutoff = NULL
)
```

**Arguments**

queryPeak	query bed file
targetPeak	target bed file(s) or folder containing bed files
TxDb	TxDb
pAdjustMethod	pvalue adjustment method
chainFile	chain file for liftOver
distanceToTSS_cutoff	restrict nearest gene annotation by distance cutoff

**Value**

data.frame

**Author(s)**

G Yu

## Examples

```
if (interactive()) {  
  require(TxDb.Hsapiens.UCSC.hg38.knownGene)  
  txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene  
  peakfile <- system.file("extdata", "demo_peak.txt", package = "epiSeeker")  
  enrichAnnoOverlap(peakfile, peakfile, txdb)  
}
```

---

enrichPeakOverlap      *enrichPeakOverlap*

---

## Description

calculate overlap significant of ChIP experiments based on the genome coordinations

## Usage

```
enrichPeakOverlap(  
  queryPeak,  
  targetPeak,  
  TxDb = NULL,  
  pAdjustMethod = "BH",  
  nShuffle = 1000,  
  chainFile = NULL,  
  pool = TRUE,  
  mc.cores = detectCores() - 1,  
  verbose = TRUE  
)
```

## Arguments

queryPeak	query bed file or GRanges object
targetPeak	target bed file(s) or folder that containing bed files or a list of GRanges objects
TxDb	TxDb
pAdjustMethod	pvalue adjustment method
nShuffle	shuffle numbers
chainFile	chain file for liftOver
pool	logical, whether pool target peaks
mc.cores	number of cores, see <a href="#">mclapply</a>
verbose	logical

## Value

data.frame

## Author(s)

G Yu

**Examples**

```
require(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
peakfile <- system.file("extdata", "demo_peak.txt", package = "epiSeeker")
peak <- readPeakFile(peakfile)[1:10]
enrichPeakOverlap(peak, peakfile, txdb, mc.cores = 1, nShuffle = 20)
```

---

epiSeekerCache	<i>Name of the epiSeeker cache environment (internal static variable)</i>
----------------	---

---

**Description**

Name of the epiSeeker cache environment (internal static variable)

**Usage**

```
epiSeekerCache
```

**Format**

character vector

---

extend_gr	<i>Extend regions functions</i>
-----------	---------------------------------

---

**Description**

Extend regions functions

**Usage**

```
extend_gr(regions, upstream, downstream, by, type)
```

**Arguments**

regions	GRanges object
upstream	upstream extension. One of actual number or rel() object.
downstream	downstream extension. One of actual number or rel() object.
by	one of 'gene', 'transcript', 'exon', 'intron', '3UTR', '5UTR', 'UTR'.
type	one of "start_site", "end_site", "body".

**Value**

GRanges object

---

filter.GRanges	<i>Extend filter to Peak (GRanges class object)</i>
----------------	---

---

**Description**

Extend filter to Peak (GRanges class object)

**Usage**

```
## S3 method for class 'GRanges'
filter(.data, ..., .by = NULL, .preserve = FALSE)
```

**Arguments**

.data	granges object
...	additional parameters
.by	Optional grouping variable(s) (column name or variable expression) specifying which columns to group by when applying filters
.preserve	Logical value indicating whether to preserve the original grouping structure when .by is specified. If TRUE, group order and identities are maintained

**Value**

A filtered GRanges object containing only rows that meet the specified criteria  
grange object

**Examples**

```
peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
peak <- readPeakFile(peakfile)
dplyr::filter(peak, fold_enrichment > 20)
```

---

getAnnoStat	<i>getAnnoStat</i>
-------------	--------------------

---

**Description**

getting status of annotation

**Usage**

```
getAnnoStat(x)
```

**Arguments**

x	csAnno object
---	---------------

**Value**

data frame

**Examples**

```
data(peakAnno)
getAnnoStat(peakAnno)
```

---

getBioRegion	<i>Prepare a bioregion of selected feature</i>
--------------	--

---

**Description**

Prepare a bioregion of selected feature

**Usage**

```
getBioRegion(
  TxDb = NULL,
  upstream = 1000,
  downstream = 1000,
  by = "gene",
  type = "start_site"
)
```

**Arguments**

TxDb	TxDb object or self-made granges object.
upstream	upstream extension. One of actual number or rel() object.
downstream	downstream extension. One of actual number or rel() object.
by	one of 'gene', 'transcript', 'exon', 'intron', '3UTR', '5UTR', 'UTR'.
type	one of "start_site", "end_site", "body".

**Details**

this function combined previous functions getPromoters(), getBioRegion() and getGeneBody() in order to solve the following issues.

(1) <<https://github.com/GuangchuangYu/ChIPseeker/issues/16>>

(2) <<https://github.com/GuangchuangYu/ChIPseeker/issues/87>>

1. function can provide a region of interest from txdb object. 2. function can make region from granges object. txdb object do not contain insulator or enhancer regions. Users can provide these regions through self-made granges object <https://github.com/YuLab-SMU/ChIPseeker/issues/189>.

There are three kinds of way to extend regions: start\_site, end\_site and body. We take transcript region to explain the differences of these three regions (tx: chr1 1000 1400).

(1) body region refers to the 1000 ~ 1400 bp.

(2) start\_site region with (upstream = upstream = 100) refers to 900-1100bp.

(3) end\_site region with (upstream = upstream = 100) refers to 1300-1500bp.

**Value**

GRanges object

**Author(s)**

Guangchuang Yu

**Examples**

```
require(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
getBioRegion(txdb)
```

---

`getBmMatrix`*getBmMatrix methods generics*

---

**Description**

getBmMatrix method for [bsseq::BSseq]

getBmMatrix method for [bmData](#)**Usage**

```
getBmMatrix(
  region,
  input,
  BSgenome,
  base = NULL,
  motif = NULL,
  position_bias = NULL,
  ...
)

## S4 method for signature 'ANY,BSseq'
getBmMatrix(
  region,
  input,
  BSgenome,
  base = NULL,
  motif = NULL,
  position_bias = NULL,
  cover_depth = TRUE,
  ...
)

## S4 method for signature 'ANY,bmData'
getBmMatrix(
  region,
  input,
  BSgenome,
  base = NULL,
  motif = NULL,
  position_bias = NULL,
  ...
)
```

**Arguments**

region	base modification region in the form of dataframe, having columns of "chr", "start" and "end"
input	the input data stored in BSseq objects or BSseqExtra objects
BSgenome	genome reference
base	one of A/T/G/C/U
motif	the motif(e.g C:CG/CH, A:GAGG/AGG) of the base modification
position_bias	1-base bias. e.g position_bias = 1("C" in "CHH"), position_bias = 2("A" in "GAGG")
...	other parameters
cover_depth	take the depth of cover into account or not

**Value**

data.frame  
dataframe

**Examples**

```
require(BSgenome.Hsapiens.UCSC.hg38)
data(demo_bmdata)
bmMatrix <- getBmMatrix(
  region = data.frame(chr = "chr22", start = 10525991, end = 10526342),
  BSgenome = BSgenome.Hsapiens.UCSC.hg38,
  input = demo_bmdata,
  base = "C",
  motif = c("CG")
)
```

---

getBmMatrix.bmData     *get the information of base modification*

---

**Description**

get the information of base modification

**Usage**

```
getBmMatrix.bmData(
  region,
  input,
  BSgenome,
  base = NULL,
  motif = NULL,
  position_bias = NULL
)
```

**Arguments**

region	base modification region in the form of dataframe, having columns of "chr", "start" and "end"
input	the input data stored in bmData objects
BSgenome	genome reference
base	one of A/T/G/C/U
motif	the motif(e.g C:CG/CH, A:GAGG/AGG) of the base modification
position_bias	1-base bias. e.g position_bias = 1("C" in "CHH"), position_bias = 2("A" in "GAGG")

**Details**

This function retrieve the information of each base, requiring bmData object as input. Then organized it to dataframe.

**Value**

dataframe

---

getBmMatrix.BSseq	<i>Get the information of base modification</i>
-------------------	---

---

**Description**

Get the information of base modification

**Usage**

```
getBmMatrix.BSseq(
  region,
  input,
  BSgenome,
  cover_depth = TRUE,
  base = NULL,
  motif = NULL,
  position_bias = NULL
)
```

**Arguments**

region	base modification region in the form of data.frame, having columns of "chr", "start" and "end"
input	the input data stored in [bsseq::BSseq] objects
BSgenome	genome reference
cover_depth	take the depth of cover into account or not
base	one of A/T/G/C/U
motif	the motif(e.g C:CG/CH, A:GAGG/AGG) of the base modification
position_bias	1-base bias. e.g position_bias = 1("C" in "CHH"), position_bias = 2("A" in "GAGG")

**Details**

This function retrieve the information of each base, requiring [bsseq::BSseq] object as input. Then organized it to data.frame.

**Value**

data.frame

---

getGeneAnno	<i>getGeneAnno</i>
-------------	--------------------

---

**Description**

Get gene annotation, symbol, gene name etc.

**Usage**

```
getGeneAnno(annoDb, geneID, type, columns)
```

**Arguments**

annoDb	annotation package
geneID	query geneID
type	gene ID type
columns	names of columns to be obtained from database

**Value**

data.frame

**Author(s)**

G Yu

---

getGenomicAnnotation	<i>getGenomicAnnotation</i>
----------------------	-----------------------------

---

**Description**

Get Genomic Annotation of peaks

**Usage**

```
getGenomicAnnotation(
  peaks,
  distance,
  tssRegion = c(-3000, 3000),
  TxDb,
  level,
  genomicAnnotationPriority,
  sameStrand = FALSE
)
```

**Arguments**

peaks	peaks in GRanges object
distance	distance of peak to TSS
tssRegion	tssRegion, default is -3kb to +3kb
TxDb	TxDb object
level	one of gene or transcript
genomicAnnotationPriority	genomic Annotation Priority
sameStrand	whether annotate gene in same strand

**Value**

character vector

**Author(s)**

G Yu

---

`getGEOgenomeVersion`    *getGEOgenomeVersion*

---

**Description**

Get genome version statistics collecting from GEO ChIPseq data

**Usage**

```
getGEOgenomeVersion()
```

**Value**

data.frame

**Author(s)**

G Yu

**Examples**

```
getGEOgenomeVersion()
```

getGEOInfo

*getGEOInfo*

---

**Description**

Get subset of GEO information by genome version keyword

**Usage**

```
getGEOInfo(genome, simplify = TRUE)
```

**Arguments**

genome	genome version
simplify	simplify result or not

**Value**

data.frame

**Author(s)**

G Yu

**Examples**

```
hg19 <- getGEOInfo(genome = "hg19", simplify = TRUE)
```

---

getGEOspecies

*getGEOspecies*

---

**Description**

Accessing species statistics collecting from GEO database

**Usage**

```
getGEOspecies()
```

**Value**

data.frame

**Author(s)**

G Yu

**Examples**

```
getGEOspecies()
```

---

getMotifMatrix                    *Get the information of motif in a range*

---

**Description**

Get the information of motif in a range

**Usage**

```
getMotifMatrix(region, pwm, ref_obj, by = "name")
```

**Arguments**

region	region object in granges.
pwm	PFMatrixList.
ref_obj	seq reference object. e.g. BSgenome object.
by	show the motif by name or ID.

**Value**

score matrix

**Examples**

```
require(BSgenome.Hsapiens.UCSC.hg38)
data(pwm_obj)

region_oi <- GRanges(
  seqnames = "chr22",
  ranges = IRanges(start = 10525891, end = 10525991)
)
motifMatrix <- getMotifMatrix(
  region = region_oi,
  pwm = pwm_obj[c(45, 120, 170)],
  ref_obj = BSgenome.Hsapiens.UCSC.hg38
)
```

---

getNearestFeatureIndicesAndDistances  
*getNearestFeatureIndicesAndDistances*

---

**Description**

Get index of features that closest to peak and calculate distance

**Usage**

```

getNearestFeatureIndicesAndDistances(
  peaks,
  features,
  sameStrand = FALSE,
  ignoreOverlap = FALSE,
  ignoreUpstream = FALSE,
  ignoreDownstream = FALSE,
  overlap = "TSS"
)

```

**Arguments**

peaks	peak in GRanges
features	features in GRanges
sameStrand	logical, whether find nearest gene in the same strand
ignoreOverlap	logical, whether ignore overlap of TSS with peak
ignoreUpstream	logical, if True only annotate gene at the 3' of the peak.
ignoreDownstream	logical, if True only annotate gene at the 5' of the peak.
overlap	one of "TSS" or "all"

**Value**

list

**Author(s)**

G Yu

---

getPromoters	<i>Get promoter region in GRanges format</i>
--------------	--

---

**Description**

Get promoter region in GRanges format

**Usage**

```
getPromoters(TxDb = NULL, upstream = 1000, downstream = 1000, by = "gene")
```

**Arguments**

TxDb	TxDb object
upstream	upstream extension. One of actual number or rel() object.
downstream	downstream extension. One of actual number or rel() object.
by	one of 'gene', 'transcript'.

**Value**

GRanges object

**Author(s)**

Guangchuang Yu

**Examples**

```
require(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
promoters <- getPromoters(TxDb = txdb, upstream = 1000, downstream = 1000)
```

---

`getSampleFiles`            *getSampleFiles*

---

**Description**

get filenames of sample files

**Usage**

```
getSampleFiles()
```

**Value**

list of file names

**Author(s)**

G Yu

**Examples**

```
files <- getSampleFiles()
```

---

`getTagMatrix`            *getTagMatrix*

---

**Description**

getTagMatrix

**Usage**

```

getTagMatrix(
  peak,
  upstream = 0,
  downstream = 0,
  windows = NULL,
  type = NULL,
  by = NULL,
  TxDb = NULL,
  weightCol = NULL,
  nbin = NULL,
  verbose = TRUE,
  ignore_strand = FALSE
)

```

**Arguments**

peak	(1) a peak file or GRanges object. (2) a list of peak file or GRanges object.
upstream	upstream extension. One of actual number or rel() object.
downstream	downstream extension. One of actual number or rel() object.
windows	a collection of region
type	one of "start_site", "end_site", "body"
by	one of 'gene', 'transcript', 'exon', 'intron', '3UTR', '5UTR', or specified by users
TxDb	TxDb or self-made granges object, served as txdb
weightCol	column name of weight, default is NULL. This column acts as a weight vaule. Details see <a href="https://github.com/YuLab-SMU/ChIPseeker/issues/15">https://github.com/YuLab-SMU/ChIPseeker/issues/15</a>
nbin	the amount of nbines. Calculate the tagMatrix by binning method. Idea is derived from the function of deeptools( <a href="https://deeptools.readthedocs.io/en/develop/content/tools/compu">https://deeptools.readthedocs.io/en/develop/content/tools/compu</a> )
verbose	print message or not
ignore_strand	ignore the strand information or not

**Details**

getTagMatrix() function can produce the matrix for visualization. Matrix represents the peak count in a windows and there are two ways to specify the 'windows':

(1) use [getPromoters](#) and [getBioRegion](#) to get 'windows' and put it into windows parameter in getTagMatrix().

(2) use getTagMatrix() to call getPromoters()/getBioRegion(). In this way users do not need to input 'windows' parameter but need to input 'TxDb' parameter. 'TxDb' can accept a set of packages contained annotation of regions of different genomes(e.g. TxDb.Hsapiens.UCSC.hg38.knownGene). Users can get the regions of interest through specific functions. These specific functions are built in getPromoters()/getBioRegion().

However, many regions can not be gain through txdb(e.g. insulator and enhancer regions), Users can provide these regions in the form of granges object. These self-made granges object will be passed to 'TxDb' and they will be passed to makeBioRegionFromGranges() to produce the 'windows'.

In a word, 'TxDb' parameter getTagMatrix() is a reference information. Users can pass txdb object or self-made granges into it.

**Value**

tagMatrix

**Author(s)**

G Yu

**Examples**

```

if (interactive()) {
  require(TxDb.Hsapiens.UCSC.hg38.knownGene)
  txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
  data(demo_peak)
  tagMatrix <- getTagMatrix(demo_peak,
    type = "start_site", by = "gene",
    upstream = 500, downstream = 500,
    TxDb = txdb, weightCol = "V7"
  )
}

```

---

getTagMatrix.internal *getTagMatrix internal function*

---

**Description**

getTagMatrix internal function

**Usage**

```

getTagMatrix.internal(
  peak,
  upstream = 0,
  downstream = 0,
  windows = NULL,
  type = NULL,
  by = NULL,
  TxDb = NULL,
  weightCol = NULL,
  nbin = NULL,
  verbose = TRUE,
  ignore_strand = FALSE
)

```

**Arguments**

peak	peak file or GRanges object
upstream	upstream extension. One of actual number or rel() object.
downstream	downstream extension. One of actual number or rel() object.
windows	a collection of region
type	one of "start_site", "end_site", "body"

by	one of 'gene', 'transcript', 'exon', 'intron', '3UTR', '5UTR', or specified by users
TxDb	TxDb or self-made granges object, served as txdb
weightCol	column name of weight, default is NULL.
nbin	the amount of nbins.
verbose	print message or not
ignore_strand	ignore the strand information or not

**Value**

matrix

---

<code>getTagMatrix_body</code>	<i>getTagMatrix function for region of body</i>
--------------------------------	---

---

**Description**

getTagMatrix function for region of body

**Usage**

```
getTagMatrix_body(
  peak.cov,
  windows,
  nbin,
  verbose = TRUE,
  ignore_strand = FALSE
)
```

**Arguments**

peak.cov	peak coverage.
windows	a collection of region.
nbin	the amount of nbins
verbose	print message or not
ignore_strand	ignore the strand information or not

**Value**

tagMatrix

---

getTagMatrix\_body\_internal  
*get tagmatrix internal function*

---

**Description**

get tagmatrix internal function

**Usage**

```
getTagMatrix_body_internal(peak.cov, windows, nbin, chr.idx)
```

**Arguments**

peak.cov	peak coverage.
windows	a collection of region.
nbin	the amount of nbines.
chr.idx	idx of chr.

**Value**

matrix

---

getTagMatrix\_site *getTagMatrix function for region of site*

---

**Description**

getTagMatrix function for region of site

**Usage**

```
getTagMatrix_site(
  peak.cov,
  windows,
  chr.idx,
  nbin = NULL,
  verbose = TRUE,
  ignore_strand = FALSE
)
```

**Arguments**

peak.cov	peak coverage.
windows	a collection of region.
chr.idx	idx of chr.
nbin	the amount of nbines
verbose	print message or not
ignore_strand	ignore the strand information or not

**Value**

tagMatrix

---

grange2mt	<i>change a list grange object to matrix</i>
-----------	--

---

**Description**

change a list grange object to matrix

**Usage**

```
grange2mt(gr_list, weightCol = NULL)
```

**Arguments**

gr_list	grange list object
weightCol	weight column of peak.

**Value**

matrix

**Examples**

```
data(demo_peak)
grange2mt(list(a = demo_peak, b = demo_peak), "V5")
```

---

gsminfo	<i>Information Datasets</i>
---------	-----------------------------

---

**Description**

ucsc genome version, precalculated data and gsm information

**Format**

A data frame with 'n' rows (GSM samples) and 14 columns.

**Value**

data frame

## Provenance

The ‘gsminfo’ dataset was constructed programmatically from public resources in the NCBI GEO and UCSC Genome Browser databases. The data generation pipeline is implemented in ‘data-raw/’ (see ‘prepareGSMInfo()’ in the package source).

Briefly, GEO metadata were retrieved using the ‘GEOmetadb’ SQLite database and ‘GEOquery’. The latest GEOmetadb SQLite file was downloaded via ‘getSQLiteFile()’ or, if unavailable, directly from <<http://starbuck1.s3.amazonaws.com/sradb/GEOmetadb.sqlite.gz>>. Platform (GPL) records were queried to identify platforms associated with high-throughput sequencing experiments. For each sequencing platform, the corresponding GSM records were obtained using ‘Meta(getGEO())’. Supplementary BED-like files for each GSM were collected using ‘getGSMsuppFile()’ and ‘batchGetGSMsuppFile()’.

Additional metadata fields (title, organism, extract protocol, characteristics, data processing description, submission date, and supplementary file URLs) were extracted from GSM SOFT files downloaded using ‘GEOquery’. Genome assembly versions for each GSM were inferred using the function ‘getGenomicVersion()’, which matches UCSC genome labels to either the data processing description or the supplementary file names, using the reference table provided in the internal dataset ‘ucsc\_release’.

PubMed IDs associated with each GEO series (GSE) were obtained from the ‘gse’ table in GEOmetadb. All GSM-level metadata were merged, cleaned, and converted to ASCII using ‘iconv()’ to remove non-ASCII characters.

Finally, newly processed GSM entries were appended to any preexisting ‘gsminfo’ object stored in the package, deduplicated, and saved as ‘gsminfo.rda’ with ‘compress="xz"’.

Thus, ‘gsminfo’ represents a curated, reproducibly constructed metadata table summarizing GEO high-throughput sequencing samples, including organism, platform, experimental descriptions, processing information, genome versions, supplementary BED file locations, and associated PubMed IDs.

## Data structure

A data frame with one row per GSM sample and the following columns:

- ‘series\_id’ GEO series accession (GSE).
- ‘gsm’ GEO sample accession (GSM).
- ‘gpl’ GEO platform accession (GPL).
- ‘organism’ Organism name (e.g., \*Mus musculus\*).
- ‘title’ Sample title as provided in GEO.
- ‘characteristics’ Experiment-specific metadata such as cell type, treatment, or antibody.
- ‘source\_name’ Source material for sequencing, typically cell or tissue type.
- ‘extract\_protocol’ Detailed wet-lab protocol for chromatin extraction, immunoprecipitation, and library preparation as reported in GEO.
- ‘description’ Antibody information or additional sample description.
- ‘data\_processing’ Bioinformatics processing description including aligner, genome build, peak calling method, and filtering steps.
- ‘submission\_date’ Date when the sample was submitted to GEO.
- ‘supplementary\_file’ URL to supplementary processed files (e.g., BED).
- ‘genomeVersion’ Genome assembly used in the processed data (e.g., mm8, hg19).
- ‘pubmed\_id’ PMID of the reference publication associated with the dataset.

---

loadTxDb	<i>load default txdb</i>
----------	--------------------------

---

**Description**

load default txdb

**Usage**

```
loadTxDb(TxDb)
```

**Arguments**

TxDb            txdb.

**Value**

txdb object

---

makeBmDataFromData	<i>makeBmDataFromData method generics</i>
--------------------	---

---

**Description**

makeBmDataFromData method generics

makeBmDataFromData method for 'CompressedGRangesList' objects

makeBmDataFromData method for 'GRanges' objects

makeBmDataFromData method for 'list' objects

makeBmDataFromData method for data.frame objects

**Usage**

```
makeBmDataFromData(data, sampleNames = NULL)
```

```
## S4 method for signature 'CompressedGRangesList'
```

```
makeBmDataFromData(data, sampleNames = NULL)
```

```
## S4 method for signature 'GRanges'
```

```
makeBmDataFromData(data, sampleNames = NULL)
```

```
## S4 method for signature 'list'
```

```
makeBmDataFromData(data, sampleNames = NULL)
```

```
## S4 method for signature 'data.frame'
```

```
makeBmDataFromData(data, sampleNames = NULL)
```

**Arguments**

data	lists object
sampleNames	the name of each samples

**Details**

The objects in 'data' must have specific forms. Columns should be features, which should be organized in the order of "chr", "pos", "value1", "value2(optional)". chr stands for chromosome. pos stands for position on chromosome, also known as coordinates. value1/2 stands for the value on each base. The colnames can be any character but must be in the order. Rows stands for each observation.

The objects in data must have specific forms. Columns should be features, which should be organized in the order of "chr", "pos", "value1", "value2(optional)". chr stands for chromosome. pos stands for position on chromosome, also known as coordinates. value1/2 stands for the value on each base. The colnames can be any character but must be in the order. Rows stands for each observation.

**Value**

bmData

**Examples**

```
demo_bisseq_file <- system.file("extdata", "demo_bisseq.txt",
  package = "epiSeeker"
)
demo_bisseq <- read.table(demo_bisseq_file, header = TRUE)
demo_bmdata <- makeBmDataFromData(
  data = list(acinar_methyl = demo_bisseq),
  sampleNames = "acinar_methyl"
)
```

---

makeBmDataFromData.internal

*makeBmDataFromData.internal*

---

**Description**

make dmData object from data

**Usage**

```
makeBmDataFromData.internal(data, sampleNames = NULL)
```

**Arguments**

data	lists object
sampleNames	the name of each samples

**Details**

This internal function was inspired by `DSS::makeBSseqData`.

The objects in data must have specific forms. Columns should be features, which should be organized in the order of "chr", "pos", "value1", "value2(optional)". chr stands for chromosome. pos stands for position on chromosome, also known as coordinates. value1/2 stands for the value on each base. The colnames can be any character but must be in the order. Rows stands for each observation.

**Value**

dmData object

---

makeBmDataFromFiles	<i>make bmData from files</i>
---------------------	-------------------------------

---

**Description**

This function makes bmData object from files. Users can input the name of a file or a file folder.

**Usage**

```
makeBmDataFromFiles(name, sampleNames = NULL, variablesNames = NULL)
```

**Arguments**

name	the name of files or file folder
sampleNames	the name for each file
variablesNames	the names of the first two columns will be assigned c("chr","pos"), the names of the following columns will be assigned by variablesNames

**Details**

bed files and txt files are supported. Bed files can only contain no more than two metadata, as it stands for value1/2. Txt files should organize the columns as chr, pos, value1, value2(optional).

**Value**

bmData

**Examples**

```
demo_bisseq_file <- system.file("extdata", "demo_bisseq.txt", package = "epiSeeker")
data <- makeBmDataFromFiles(demo_bisseq_file,
  sampleNames = "acinar_methyl",
  variablesNames = c("Cov", "Methylation")
)
```

---

mutate.GRanges	<i>Extend mutate to Peak (GRanges class object)</i>
----------------	---

---

## Description

Extend mutate to Peak (GRanges class object)

## Usage

```
## S3 method for class 'GRanges'  
mutate(  
  .data,  
  ...,  
  .by = NULL,  
  .keep = c("all", "used", "unused", "none"),  
  .before = NULL,  
  .after = NULL  
)
```

## Arguments

<code>.data</code>	granges object
<code>...</code>	additional parameters
<code>.by</code>	Optional grouping variable(s) (column name or variable expression) specifying which columns to group by for operations
<code>.keep</code>	Character vector specifying which columns to retain. Possible values: "all" (retain all columns, default), "used" (retain only columns used in calculations), "unused" (retain only columns not used in calculations), "none" (retain only newly created columns)
<code>.before</code>	Column name or position index specifying where to insert new columns before
<code>.after</code>	Column name or position index specifying where to insert new columns after

## Value

A processed GRanges object containing the added or modified columns

## Examples

```
peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")  
peak <- readPeakFile(peakfile)  
dplyr::mutate(peak, score = tags)
```

overlap                      *overlap*

---

**Description**

calculate the overlap matrix, which is useful for vennplot

**Usage**

```
overlap(Sets)
```

**Arguments**

Sets                      a list of objects

**Value**

data.frame

**Author(s)**

G Yu

---

parse\_peak                      *parse peak str*

---

**Description**

parse peak str

**Usage**

```
parse_peak(peak_str)
```

**Arguments**

peak\_str                      peak str

**Value**

data frame

**Examples**

```
parse_peak("chr1:150235946-150236624")
```

---

peakAnno

*Example data of peak annotation*

---

### Description

A 'csAnno' object representing the annotation result of the example peak set 'demo\_peak'. Peaks were annotated using the function 'annotateSeq()' in 'epiSeeker'.

### Format

A 'csAnno' object containing 220 annotated peaks.

### Value

csAnno object

### Provenance

Input peaks were taken from the example dataset 'demo\_peak'. Annotation was generated using 'epiSeeker::annotateSeq()'.

### Data structure

A 'csAnno' S4 object with the following slots:

**'anno'** A 'GRanges' object containing the annotated peaks, including peak coordinates, basic peak metrics, and gene-based annotation fields.

**'tssRegion'** Numeric vector of length two defining the upstream and downstream window used for TSS annotation.

**'level'** Character string indicating whether annotation was performed at the "transcript" or "gene" level.

**'hasGenomicAnnotation'** Logical value indicating whether detailed genomic annotation (promoter, exon, intron, etc.) was computed.

**'detailGenomicAnnotation'** A data frame providing per-peak binary indicators for genomic categories.

**'annoStat'** A data frame summarizing annotation category frequencies across the annotated peak set.

**'peakNum'** Total number of annotated peaks.

---

peakAnnoList	<i>Example data of a list of peak annotation</i>
--------------	--

---

**Description**

A list of csAnno objects obtained by annotating multiple peak files using `epiSeeker::annotateSeq()`. See `data-raw/example_data.R`

**Format**

A a list of csAnno objects.

**Value**

list of csAnno object

**Provenance**

The example peak annotation list was generated using several example peak files returned by `getSampleFiles()`. Each peak file was annotated using `epiSeeker::annotateSeq()`.

**Data structure**

A named list where each element is a csAnno S4 object produced by `annotateSeq()`.

---

plotAnnoBar	<i>plotAnnoBar method generics</i>
-------------	------------------------------------

---

**Description**

plotAnnoBar method for 'csAnno' instance

**Usage**

```
plotAnnoBar(
  x,
  xlab = "",
  ylab = "Percentage%",
  title = "Feature Distribution",
  ...
)

## S4 method for signature 'list'
plotAnnoBar(
  x,
  xlab = "",
  ylab = "Percentage%",
  title = "Feature Distribution",
  ...
)

plotAnnoBar(x, xlab="", ylab='Percentage%',title="Feature Distribution", ...)
```

**Arguments**

x	'csAnno' instance
xlab	xlab
ylab	ylab
title	title
...	additional paramter

**Value**

plot

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
data(peakAnno)
plotAnnoBar(peakAnno)
```

---

`plotAnnoBar.data.frame`  
*plotAnnoBar.data.frame*

---

**Description**

Plot feature distribution based on their chromosome region

**Usage**

```
plotAnnoBar.data.frame(  
  anno.df,  
  xlab = "",  
  ylab = "Percentage%",  
  title = "Feature Distribution",  
  categoryColumn  
)
```

**Arguments**

anno.df	annotation stats
xlab	xlab
ylab	ylab
title	plot title
categoryColumn	category column

**Details**

plot chromosome region features

**Value**

bar plot that summarize genomic features of peaks

**Author(s)**

Guangchuang Yu <<https://yulab-smu.top>>

**See Also**

[[annotateSeq\(\)](#)] [[plotAnnoPie\(\)](#)]

---

plotAnnoPie

*plotAnnoPie method generics*

---

**Description**

plotAnnoPie method for 'csAnno' instance

**Usage**

```
plotAnnoPie(
  x,
  ndigit = 2,
  cex = 0.9,
  col = NA,
  legend.position = "rightside",
  pie3D = FALSE,
  radius = 0.8,
  ...
)
```

```
plotAnnoPie(x, ndigit = 2, cex = 0.9, col = NA,
  legend.position = "rightside", pie3D = FALSE,
  radius = 0.8, ...)
```

**Arguments**

x	'csAnno' instance
ndigit	number of digit to round
cex	label cex
col	color
legend.position	topright or other.
pie3D	plot in 3D or not
radius	radius of the pie
...	extra parameter

**Value**

plot

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
data(peakAnno)
plotAnnoPie(peakAnno)
```

---

plotAnnoPie.csAnno      *plotAnnoPie*

---

**Description**

pieplot from peak genomic annotation

**Usage**

```
plotAnnoPie.csAnno(
  x,
  ndigit = 2,
  cex = 0.8,
  col = NA,
  legend.position = "rightside",
  pie3D = FALSE,
  radius = 0.8,
  ...
)
```

**Arguments**

x	csAnno object
ndigit	number of digit to round
cex	label cex
col	color
legend.position	topright or other.
pie3D	plot in 3D or not
radius	radius of Pie
...	extra parameter

**Value**

pie plot of peak genomic feature annotation

**Author(s)**

Guangchuang Yu <<https://yulab-smu.top>>

**See Also**

[annotateSeq()] [plotAnnoBar()]

**Examples**

```
data(peakAnno)
plotAnnoPie(peakAnno)
```

---

plotBmProf

*plotBmProf*

---

**Description**

Plot base modification profile

**Usage**

```
plotBmProf(
  df,
  motif_color = NULL,
  title = NULL,
  xlim = NULL,
  interactive = FALSE,
  width_svg = 10,
  height_svg = 6,
  highlight = NULL,
  highlight_color = "#c6c3c3",
  highlight_alpha = 0.2,
  xlab = "Genomic Region(5'->3')",
  ylab = NULL,
  second_ylab = NULL,
  switch_y_value = TRUE,
  legend_lab_motif = NULL,
  legend_lab_value2 = NULL,
  strip_placement = "outside",
  angle_of_facet_label = 360,
  alpha = 0.6,
  y_ticks_length = 0.25,
  x_ticks_length = 0.25,
  auto_x_axis = TRUE,
  strip_border = FALSE,
  facet_label_text_size = 10,
  axis_title_text_size = 17,
  title_text_size = 20,
  right_y_axis_text_size = 10,
  left_y_axis_text_size = 10,
  x_axis_text_size = 10,
  depth_heatmap = TRUE,
  nrow = NULL,
  ncol = NULL,
  panel_spacing = 1,
```

```

    legend_box_spacing = 3,
    legend_position = "right"
)

```

### Arguments

df	the base modification data.frame
motif_color	the color for different motifs(CHH,CHG,CG)
title	the title of the plot, can also be a list of title.
xlim	the specified interval of region, must be the sub-interval of the dmR. list for list df
interactive	produce interactive fig or not.
width_svg	width_svg.
height_svg	height_svg.
highlight	a region or a list of region to highlight.
highlight_color	colors of highlight rect. Default "#c6c3c3"
highlight_alpha	alpha of highlight rect.
xlab	the x label, can also be a list of x label
ylab	the y label, can also be a list of y label
second_ylab	the ylab for second y-axis
switch_y_value	switch the value from left y-axis to right y-axis
legend_lab_motif	the label of legend for motif
legend_lab_value2	the label of legend for the second value(ylab is the label for the first value)
strip_placement	strip.placement
angle_of_facet_label	the angle of facet label, e.g. 0 is horizontal
alpha	transparency for the depth information line
y_ticks_length	the length of y-axis ticks
x_ticks_length	the length of x-axis ticks
auto_x_axis	use auto x axis or not.
strip_border	add border to the facet label or not
facet_label_text_size	the size of facet label text
axis_title_text_size	the size of axis title text
title_text_size	the size of the title text
right_y_axis_text_size	the size of the left y axis text,this work when depth information is taken into account
left_y_axis_text_size	the size of the left y axis text

`x_axis_text_size` the size of x axis text  
`depth_heatmap` draw the heatmap of depth information or not  
`nrow` the nrow of plotting a list of dmR  
`ncol` the ncol of plotting a list of dmR  
`panel_spacing` the distance between panels  
`legend_box_spacing` the distance between legend and plotting area,"cm"  
`legend_position` the position of legend

**Value**

ggplot object

**Examples**

```

require(BSgenome.Hsapiens.UCSC.hg38)
data(demo_bmdata)
bmMatrix <- getBmMatrix(
  region = data.frame(chr = "chr22", start = 10525991, end = 10526342),
  BSgenome = BSgenome.Hsapiens.UCSC.hg38,
  input = demo_bmdata,
  # base = "C",
  motif = c("CG")
)
plotBmProf(bmMatrix)
  
```

---

plotCov

*plotCov*

---

**Description**

plotCov

**Usage**

```

plotCov(
  peak,
  weightCol = NULL,
  facet_level = NULL,
  highlight = NULL,
  highlight_color = "#c6c3c3",
  highlight_alpha = 0.2,
  xlab = "Chromosome Size (bp)",
  ylab = "",
  interactive = FALSE,
  width_svg = 10,
  height_svg = 6,
  title = "ChIP Peaks over Chromosomes",
  x_text_size = 10,
  
```

```

y_text_size = 10,
facet_label_text_size = 10,
chrs = NULL,
xlim = NULL,
facet_var = NULL,
facet_scales = "free",
lower = 1,
fill_color = "black",
add_cluster_tree = FALSE,
cluster_dist_method = "euclidean",
cluster_hclust_method = "complete",
legend_position = NULL,
add_coaccess = FALSE,
curvature = 0.3,
coaccess_top_n = NULL,
coaccess_cor_threshold = NULL,
design = NULL,
coaccess_legend_pos = c(0.9, 0.5),
coaccess_legend_text_size = 10,
coaccess_legend_title_size = 12
)

```

### Arguments

peak	peak file or GRanges object.
weightCol	weight column of peak.
facet_level	facet_level.
highlight	a region or a list of region to highlight.
highlight_color	colors of highlight rect. Default "#c6c3c3"
highlight_alpha	alpha of highlight rect.
xlab	xlab.
ylab	ylab.
interactive	produce interactive fig or not.
width_svg	width_svg
height_svg	height_svg
title	title.
x_text_size	the size of x text.
y_text_size	the size of y text.
facet_label_text_size	the size of facet label text.
chrs	selected chromosomes to plot, all chromosomes by default.
xlim	ranges to plot, default is whole chromosome.
facet_var	how to facet. one of c("chr~.", ".~ chr", ".~.id", ".id~.", ".id~chr", "chr~.id")
facet_scales	how to scale facet data. Default: "free".
lower	lower cutoff of coverage signal.

`fill_color` specify the color/palette for the plot. Order matters.  
`add_cluster_tree` add cluster tree for samples or not.  
`cluster_dist_method` method for calculate cluster tree. Details see [stats::dist()]  
`cluster_hclust_method` method for hclust. Details see [stats::hclust()]  
`legend_position` legend\_position  
`add_coaccess` add co-accessibility or not  
`curvature` curvature.  
`coaccess_top_n` top n co-accessibility to show, default: 3.  
`coaccess_cor_threshold` co-access peak cor threshold.  
`design` the design layout of figure.  
`coaccess_legend_pos` the legend position of co-accessibility plot legend.  
`coaccess_legend_text_size` the legend position of co-accessibility plot legend text size.  
`coaccess_legend_title_size` the legend position of co-accessibility plot legend title size.

### Details

Plot peak coverage

### Value

ggplot2 object

### Author(s)

G Yu

### Examples

```

peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
peak <- readPeakFile(peakfile)
plotCov(peak)
  
```

---

plotDistToTSS                    *plotDistToTSS method generics*

---

### Description

plotDistToTSS method for 'csAnno' instance

### Usage

```
plotDistToTSS(
  x,
  distanceColumn = "distanceToTSS",
  xlab = "",
  ylab = "Binding sites (%) (5'→3')",
  title = "Distribution of transcription factor-binding loci relative to TSS",
  ...
)

## S4 method for signature 'list'
plotDistToTSS(
  x,
  distanceColumn = "distanceToTSS",
  xlab = "",
  ylab = "Binding sites (%) (5'→3')",
  title = "Distribution of transcription factor-binding loci relative to TSS",
  distanceBreaks = c(0, 1000, 3000, 5000, 10000, 1e+05),
  palette = NULL,
  ...
)

plotDistToTSS(x,distanceColumn="distanceToTSS", xlab="",
ylab="Binding sites (%) (5'→3')",
title="Distribution of transcription factor-binding loci relative to TSS",...)
```

### Arguments

x	'csAnno' instance
distanceColumn	distance column name
xlab	xlab
ylab	ylab
title	title
...	additional parameter
distanceBreaks	breaks of distance, default is 'c(0, 1000, 3000, 5000, 10000, 100000)'
palette	palette name for coloring different distances. Run 'RColorBrewer::display.brewer.all()'
	to see all applicable values.

### Value

plot

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
data(peakAnno)
plotDistToTSS(peakAnno)
```

---

```
plotDistToTSS.data.frame
```

```
plotDistToTSS.data.frame
```

---

**Description**

Plot feature distribution based on the distances to the TSS

**Usage**

```
plotDistToTSS.data.frame(  
  peakDist,  
  distanceColumn = "distanceToTSS",  
  distanceBreaks = c(0, 1000, 3000, 5000, 10000, 1e+05),  
  palette = NULL,  
  xlab = "",  
  ylab = "Binding sites (%) (5'->3')",  
  title = "Distribution of transcription factor-binding loci relative to TSS",  
  categoryColumn = ".id"  
)
```

**Arguments**

<code>peakDist</code>	peak annotation
<code>distanceColumn</code>	column name of the distance from peak to nearest gene
<code>distanceBreaks</code>	default is 'c(0, 1000, 3000, 5000, 10000, 100000)'
<code>palette</code>	palette name for coloring different distances. Run 'RColorBrewer::display.brewer.all()' to see all applicable values.
<code>xlab</code>	x label
<code>ylab</code>	y label
<code>title</code>	figure title
<code>categoryColumn</code>	category column, default is ".id"

**Value**

bar plot that summarize distance from peak to TSS of the nearest gene.

**Author(s)**

Guangchuang Yu <https://guangchuangyu.github.io>

**See Also**[annotateSeq](#)


---

plotGeneTrack	<i>Plot gene track</i>
---------------	------------------------

---

**Description**

Plot gene track

**Usage**

```
plotGeneTrack(
  txdb,
  chr,
  start_pos,
  end_pos,
  xlab = "",
  ylab = "",
  x_text_size = 10,
  y_text_size = 10,
  select_gene = "all",
  palette = NULL,
  fromType = "ENTREZID",
  highlight = NULL,
  highlight_color = "#c6c3c3",
  highlight_alpha = 0.2,
  OrgDb = NULL,
  show_legend = FALSE,
  auto_x_axis = TRUE
)
```

**Arguments**

txdb	TxDb object, providing gene annotation.
chr	chromosome id.
start_pos	start coordinate of windows.
end_pos	end coordinate of windows.
xlab	x lab.
ylab	y lab.
x_text_size	the size of x text.
y_text_size	the size of y text.
select_gene	show all gene or specific gene. (1)"all", show all genes. (2) gene symbol, e.g. c("SKAP1", "EFCAB13"). (3) gene id, e.g. c(4831, 55316)
palette	palette, default "Set3".
fromType	from which type of gene name to change gene id. Default: ENTREZID. See [clusterProfiler::bitr()]

highlight	a region or a list of region to highlight.
highlight_color	colors of highlight rect. Default "#c6c3c3"
highlight_alpha	alpha of highlight rect.
OrgDb	OrgDb for change gene id to gene symbol.
show_legend	show legend or not.
auto_x_axis	use auto x axis or not.

**Value**

ggplot object

**Examples**

```
require(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
plotGeneTrack(txdb = txdb, chr = "chr8", start_pos = 126712193, end_pos = 126712293)
```

---

plotMotifProf	<i>Plot the profile of motif of specific peak</i>
---------------	---

---

**Description**

Plot the profile of motif of specific peak

**Usage**

```
plotMotifProf(
  df,
  legend_lab = "motif",
  y_lab = "motif score",
  x_lab = NULL,
  interactive = FALSE,
  width_svg = 10,
  height_svg = 6
)
```

**Arguments**

df	motif information data.frame.
legend_lab	legend lab.
y_lab	y axis label.
x_lab	x axis label.
interactive	produce interactive fig or not.
width_svg	width_svg
height_svg	height_svg

**Value**

ggplot object

**Examples**

```
require(BSgenome.Hsapiens.UCSC.hg38)
data(pwm_obj)
region_oi <- GRanges(
  seqnames = "chr22",
  ranges = IRanges(start = 10525891, end = 10525991)
)
motifMatrix <- getMotifMatrix(
  region = region_oi,
  pwm = pwm_obj[c(45, 120, 170)],
  ref_obj = BSgenome.Hsapiens.UCSC.hg38
)
plotMotifProf(motifMatrix)
```

---

plotPeakHeatmap

*plotPeakHeatmap* function

---

**Description**

plotPeakHeatmap function

**Usage**

```
plotPeakHeatmap(
  tagMatrix,
  plot_prof = TRUE,
  xlab = "",
  ylab = "",
  palette = NULL,
  title = NULL,
  facet_label_text_size = 12,
  nrow = NULL,
  ncol = NULL,
  conf = NULL,
  statistic_method = "mean",
  missingDataAsZero = TRUE,
  facet = "none",
  free_y = TRUE,
  height_proportion = 4,
  ...
)
```

**Arguments**

tagMatrix	output from getTagMatrix().
plot_prof	combine prof or not. Default: TRUE
xlab	xlab.

ylab	ylab.
palette	palette to be filled in,details see <a href="#">scale_colour_brewer</a> .
title	title.
facet_label_text_size	the size of facet label text
nrow	nrow to place a number of fig.
ncol	ncol to place a number of fig.
conf	confidence interval.
statistic_method	method to do statistic. one of "mean", "median", "min", "max", "sum", "std"
missingDataAsZero	set missing data as zero or not.
facet	one of 'none', 'row' and 'column'.
free_y	if TRUE, y will be scaled.
height_proportion	the proportion of profiling picture and heatmap
...	additional parameters

**Value**

ggplot object

**Examples**

```
data(tagMatrix)
plotPeakHeatmap(tagMatrix)
```

---

plotPeakHeatmap\_sub     *Plot peak heatmap sub function*

---

**Description**

Plot peak heatmap sub function

**Usage**

```
plotPeakHeatmap_sub(
  tagMatrix,
  xlab = "",
  ylab = "",
  palette = NULL,
  title = NULL,
  facet_label_text_size = 12,
  nrow = NULL,
  ncol = NULL
)
```

**Arguments**

tagMatrix	output from getTagMatrix().
xlab	xlab.
ylab	ylab.
palette	palette to be filled in,details see [ggplot2::scale_colour_brewer()].
title	title.
facet_label_text_size	the size of facet label text
nrow	nrow to place a number of fig.
ncol	ncol to place a number of fig.

**Value**

ggplot object

---

plotPeakHeatmap\_sub.internal  
*internal function of plotPeakHeatmap*

---

**Description**

internal function of plotPeakHeatmap

**Usage**

```
plotPeakHeatmap_sub.internal(
  tagMatrix,
  xlab = "",
  ylab = "",
  palette = NULL,
  title = NULL,
  facet_label_text_size = 12
)
```

**Arguments**

tagMatrix	output from getTagMatrix().
xlab	xlab.
ylab	ylab.
palette	palette to be filled in,details see [ggplot2::scale_colour_brewer()].
title	title.
facet_label_text_size	the size of facet label text

**Value**

ggplot object

---

plotPeakProf                    *plot peak profile*

---

### Description

plot peak profile

### Usage

```
plotPeakProf(  
  tagMatrix,  
  xlab = "Genomic Region (5'→3')",  
  ylab = "Peak Count Frequency",  
  conf = NULL,  
  title = "",  
  facet = "none",  
  free_y = TRUE,  
  statistic_method = "mean",  
  missingDataAsZero = TRUE,  
  ...  
)
```

### Arguments

tagMatrix	output from getTagMatrix().
xlab	xlab.
ylab	ylab.
conf	confidence interval.
title	title.
facet	one of 'none', 'row' and 'column'.
free_y	if TRUE, y will be scaled.
statistic_method	method to do statistic. one of "mean", "median", "min", "max", "sum", "std"
missingDataAsZero	set missing data as zero or not.
...	additional parameters

### Value

ggplot object

### Author(s)

G Yu; Y Yan

### Examples

```
data(tagMatrix)  
plotPeakProf(tagMatrix)
```

pwm\_obj

*motif reference for Homo sapiens***Description**

A collection of transcription factor position weight matrices (PWMs) retrieved from the JASPAR 2024 database. This dataset is used to demonstrate motif enrichment, motif scanning, and peak–motif association analyses in **epiSeeker**. See `data-raw/example_data.R`

**Format**

A `PFMatrixList` object containing PWMs for multiple human transcription factors from the JASPAR 2024 CORE collection.

**Value**

pwm\_obj

**Provenance**

The PWM set was obtained using the JASPAR 2024 SQLite database bundled in the **JASPAR2024** package. Matrices were retrieved using **TFBSTools** with the following parameters:

- `collection = "CORE"`
- `all_versions = FALSE`
- `species = "Homo sapiens"`
- `tax_group = "vertebrates"`

**Data structure**

A `TFBSTools::PWMMatrixList` (or `PFMatrixList`) object containing one PWM per transcription factor. Each matrix stores nucleotide position weights across the TF binding motif, with rows representing A, C, G, T and columns representing motif positions.

readPeakFile

*readPeakFile***Description**

Read peak file and store in `data.frame` or `GRanges` object

**Usage**

```
readPeakFile(peakfile, as = "GRanges", ...)
```

**Arguments**

peakfile	peak file
as	output format, one of <code>GRanges</code> or <code>data.frame</code>
...	additional parameter (pass to <code>'utils::read.delim()'</code> )

**Value**

peak information, in GRanges or data.frame object

**Author(s)**

G Yu

**Examples**

```
peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
peak.gr <- readPeakFile(peakfile, as = "GRanges")
peak.gr
```

---

reexports

*Objects exported from other packages*

---

**Description**

These objects are imported from other packages. Follow the links below to see their documentation.

**ggplot2** [rel](#)

**Value**

function

**Examples**

```
rel(0.2)
```

---

rename.GRanges

*Rename columns of a GRanges object*

---

**Description**

Rename columns of a GRanges object

**Usage**

```
## S3 method for class 'GRanges'
rename(.data, ...)
```

**Arguments**

.data            A GRanges object.  
...              Rename expressions in the form new\_name = old\_name.

**Value**

A GRanges object with renamed metadata columns.

**Examples**

```
peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
peak <- readPeakFile(peakfile)
dplyr::rename(peak, tag = tags)
```

---

seq2gene

*seq2gene*

---

**Description**

Annotate genomic regions to genes in many-to-many mapping

**Usage**

```
seq2gene(seq, tssRegion, flankDistance, TxDb, sameStrand = FALSE)
```

**Arguments**

seq	genomic regions in GRanges object
tssRegion	TSS region
flankDistance	flanking search radius
TxDb	TxDb object
sameStrand	logical, whether find nearest/overlap gene in the same strand

**Details**

This function associates genomic regions with coding genes in a many-to-many mapping. It first maps genomic regions to host genes (either located in exon or intron), proximal genes (located in promoter regions) and flanking genes (located in upstream and downstream within user-specified distance).

**Value**

gene vector

**Author(s)**

Guangchuang Yu

**Examples**

```
data(seq2gene_result)
seq2gene_result
```

---

seq2gene\_result      *Result of seq2gene*

---

### Description

A character vector of gene IDs returned by `seq2gene()`, representing genes associated with a subset of peaks. This dataset is used to illustrate peak-to-gene mapping and regulatory region annotation workflows in **epiSeeker**. See `data-raw/example_data.R`

### Format

A character vector of gene IDs generated by `seq2gene()` from the subset of peaks derived from `demo_peak`.

### Value

vector of gene names

### Data structure

A character vector of gene identifiers (ENTREZ IDs) representing genes linked to the example peak set via TSS proximity or flanking-gene search.

### Provenance

The example peak set `demo_peak` was constructed by sampling up to 10 peaks per autosome (chr1–chr22) from the ChIP-seq dataset GSM6418464. Peaks were imported using `readPeakFile()`, subset by chromosome, and combined into a single `GRanges` object.

The gene-level associations were then computed directly using:

```
seq2gene_result <- seq2gene(  
  demo_peak,  
  tssRegion = c(-1000, 1000),  
  flankDistance = 3000,  
  txdb  
)
```

The resulting character vector of gene IDs was saved via `data-raw/example_data.R`.

---

show      *show method*

---

### Description

show method for 'csAnno' instance

### Usage

```
show(object)
```

**Arguments**

object            A 'csAnno' instance

**Value**

message

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
peakfile <- system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
show(peakfile)
```

---

shuffle	<i>shuffle</i>
---------	----------------

---

**Description**

shuffle the position of peak

**Usage**

```
shuffle(peak.gr, TxDb)
```

**Arguments**

peak.gr            GRanges object  
TxDb                TxDb

**Value**

GRanges object

**Author(s)**

G Yu

**Examples**

```
require(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
p <- GRanges(
  seqnames = c("chr1", "chr3"),
  ranges = IRanges(start = c(1, 100), end = c(50, 130))
)
shuffle(p, TxDb = txdb)
```

tagMatrix

*Example data of tagMatrix***Description**

tagMatrix result used to demonstrate TSS enrichment visualization and tag distribution plotting functions in **epiSeeker**. See data-raw/example\_data.R

**Format**

A numeric matrix with n genes × 500 bins.

**Value**

matrix

**Provenance**

The tag matrix was generated using a sample peak file obtained from `getSampleFiles()[[4]]`. Peaks were imported via `readPeakFile()` and processed using `epiSeeker::getTagMatrix()` with the following settings:

- Transcript database: `Txdb.Hsapiens.UCSC.hg19.knownGene`
- Annotation mode: `type = "start_site", by = "gene"`
- TSS window: upstream 3000 bp, downstream 3000 bp
- Peak weight: column "V5" of the peak file
- Number of bins: `nbin = 500`

**Data structure**

A numeric matrix in which:

**Rows** Represent individual genes contributing tags around their TSS.

**Columns** Represent evenly spaced bins across the TSS window from -3000 bp to +3000 bp (500 bins total).

upsetplot

*upsetplot method***Description**

upsetplot method generics

**Usage**

```
upsetplot(x, ...)
```

**Arguments**

x                    A 'csAnno' instance  
...                    additional parameter

**Value**

plot

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
data(peakAnno)
upsetplot(peakAnno)
```

---

vennpie

*vennpie method generics*

---

**Description**

vennpie method generics

**Usage**

```
vennpie(x, r = 0.2, cex = 1.2, ...)
```

```
vennpie(x, r = 0.2, cex=1.2, ...)
```

**Arguments**

x                    A 'csAnno' instance  
r                    initial radius  
cex                  value to adjust legend  
...                    additional parameter

**Value**

plot

**Author(s)**

Guangchuang Yu <<https://guangchuangyu.github.io>>

**Examples**

```
data(peakAnno)
vennpie(peakAnno)
```

---

 vennplot

*vennplot*


---

**Description**

Plot the overlap of a list of object

**Usage**

```
vennplot(Sets, ...)
```

**Arguments**

Sets            a list of object, can be vector or GRanges object.

...            extra parameters using ggVennDiagram. Details see [ggVennDiagram::ggVennDiagram]

**Details**

venn plot produced through this way has colors which can be defined by users using ggplot2 grammar e.g.(scale\_fill\_distiller()). And users can specify any details, like digital number, text size and showing percentage or not, by inputting ‘...’ extra parameters.

**Value**

venn plot that summarize the overlap of peaks from different experiments or gene annotation from different peak files.

**Author(s)**

G Yu

**Examples**

```
data(peakAnnoList)
genes <- lapply(peakAnnoList, function(i) as.data.frame(i)$geneId)
vennplot(genes)
```

---

 vennplot.peakfile

*vennplot.peakfile*


---

**Description**

Vennplot for peak files

**Usage**

```
vennplot.peakfile(files, labels = NULL)
```

**Arguments**

files	peak files
labels	labels for peak files

**Value**

figure

**Author(s)**

G Yu

**Examples**

```
files <- list(
  system.file("extdata", "sample_peaks.txt", package = "epiSeeker"),
  system.file("extdata", "sample_peaks.txt", package = "epiSeeker")
)
vennplot.peakfile(files)
```

# Index

- \* **classes**
  - bmData-class, 11
  - csAnno-class, 14
- \* **datasets**
  - epiSeekerCache, 20
  - gsminfo, 36
- \* **internal**
  - epiSeeker-package, 4
  - reexports, 62
- ., 5
- .epiSeekerEnv, 5
- annotateSeq, 6, 55
- arrange.GRanges, 8
- as.data.frame.csAnno, 8
- as.GRanges, 9
  
- bin\_vector, 9
- bmData, 10, 23
- bmData-class, 11
- bmData-methods (getBmMatrix), 23
- BSseq-methods (getBmMatrix), 23
  
- check\_bin, 11
- check\_extension, 12
- check\_windows, 12
- combine\_csAnno, 13
- create\_regex\_patterns\_negative, 13
- create\_regex\_patterns\_positive, 14
- csAnno-class, 14
  
- demo\_bmdata, 15
- demo\_peak, 16
- downloadGEObedFiles, 16
- downloadGSMbedFiles, 17
- dropAnno, 17
  
- enrichAnnoOverlap, 18
- enrichPeakOverlap, 19
- epiSeeker (epiSeeker-package), 4
- epiSeeker-package, 4
- epiSeekerCache, 20
- extend\_gr, 20
  
- filter.GRanges, 21
  
- getAnnoStat, 21
- getBioRegion, 22, 32
- getBmMatrix, 23
- getBmMatrix, (getBmMatrix), 23
- getBmMatrix, ANY, bmData-method (getBmMatrix), 23
- getBmMatrix, ANY, BSseq-method (getBmMatrix), 23
- getBmMatrix.bmData, 24
- getBmMatrix.BSseq, 25
- getGeneAnno, 26
- getGenomicAnnotation, 26
- getGEOgenomeVersion, 27
- getGEOInfo, 28
- getGEOspecies, 28
- getMotifMatrix, 29
- getNearestFeatureIndicesAndDistances, 29
- getPromoters, 30, 32
- getSampleFiles, 31
- getTagMatrix, 31
- getTagMatrix.internal, 33
- getTagMatrix\_body, 34
- getTagMatrix\_body\_internal, 35
- getTagMatrix\_site, 35
- grange2mt, 36
- gsminfo, 36
  
- loadTxDb, 38
  
- makeBmDataFromData, 38
- makeBmDataFromData, CompressedGRangesList-method (makeBmDataFromData), 38
- makeBmDataFromData, data.frame-method (makeBmDataFromData), 38
- makeBmDataFromData, GRanges-method (makeBmDataFromData), 38
- makeBmDataFromData, list-method (makeBmDataFromData), 38
- makeBmDataFromData.internal, 39
- makeBmDataFromFiles, 40
- mclapply, 19
- mutate.GRanges, 41

- overlap, 42
- parse\_peak, 42
- peakAnno, 43
- peakAnnoList, 44
- plotAnnoBar, 44
- plotAnnoBar, csAnno, ANY-method (plotAnnoBar), 44
- plotAnnoBar, csAnno-method (csAnno-class), 14
- plotAnnoBar, list-method (plotAnnoBar), 44
- plotAnnoBar.data.frame, 45
- plotAnnoPie, 46
- plotAnnoPie, csAnno, ANY-method (plotAnnoPie), 46
- plotAnnoPie, csAnno-method (csAnno-class), 14
- plotAnnoPie.csAnno, 47
- plotBmProf, 48
- plotCov, 50
- plotDistToTSS, 53
- plotDistToTSS, csAnno, ANY-method (plotDistToTSS), 53
- plotDistToTSS, csAnno-method (csAnno-class), 14
- plotDistToTSS, list-method (plotDistToTSS), 53
- plotDistToTSS.data.frame, 54
- plotGeneTrack, 55
- plotMotifProf, 56
- plotPeakHeatmap, 57
- plotPeakHeatmap\_sub, 58
- plotPeakHeatmap\_sub.internal, 59
- plotPeakProf, 60
- pwm\_obj, 61
  
- readPeakFile, 61
- reexports, 62
- rel, 62
- rel (reexports), 62
- rename.GRanges, 62
  
- scale\_colour\_brewer, 58
- seq2gene, 63
- seq2gene\_result, 64
- show, 64
- show, csAnno, ANY-method (show), 64
- show, csAnno-method (csAnno-class), 14
- shuffle, 65
- subset, csAnno-method (csAnno-class), 14
  
- tagMatrix, 66
  
- ucsc\_release (gsminfo), 36
- upsetplot, 66
- upsetplot, csAnno-method (csAnno-class), 14
  
- vennpie, 67
- vennpie, csAnno-method (csAnno-class), 14
- vennplot, 68
- vennplot.peakfile, 68