

Package ‘ChIPpeakAnno’

October 8, 2015

Type Package

Title Batch annotation of the peaks identified from either ChIP-seq, ChIP-chip experiments or any experiments resulted in large number of chromosome ranges.

Version 3.2.2

Date 2015-03-12

Author Lihua Julie Zhu, Jianhong Ou, Herve Pages, Claude Gazin, Nathan Lawson, Ryan Thompson, Simon Lin, David Lapointe and Michael Green

Maintainer

Lihua Julie Zhu <julie.zhu@umassmed.edu>, Jianhong Ou <Jianhong.ou@umassmed.edu>

Depends R (>= 2.10), grid, VennDiagram, biomaRt, IRanges, Biostrings, GenomicRanges

Imports BiocGenerics (>= 0.1.0), GO.db, BSgenome, GenomicFeatures, AnnotationDbi, limma, multtest, RBGL, graph, BiocInstaller, stats

Suggests reactome.db, BSgenome.Ecoli.NCBI.20080805, org.Ce.eg.db, org.Hs.eg.db, BSgenome.Celegans.UCSC.ce10, BSgenome.Drerio.UCSC.danRer7, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, gplots, RUnit, BiocStyle, rtracklayer

Description The package includes functions to retrieve the sequences around the peak, obtain enriched Gene Ontology (GO) terms, find the nearest gene, exon, miRNA or custom features such as most conserved elements and other transcription factor binding sites supplied by users. Starting 2.0.5, new functions have been added for finding the peaks with bi-directional promoters with summary statistics (peaksNearBDP), for summarizing the occurrence of motifs in peaks (summarizePatternInPeaks) and for adding other IDs to annotated peaks or enrichedGO (addGeneIDs). This package leverages the biomaRt, IRanges, Biostrings, BSgenome, GO.db, multtest and stat packages

License GPL (>= 2)

LazyLoad yes

biocViews Annotation, ChIPSeq, ChIPchip

NeedsCompilation no

R topics documented:

ChIPpeakAnno-package	3
addAncestors	4
addGeneIDs	5
annotatedPeak	7
annotatePeakInBatch	8
assignChromosomeRegion	11
BED2RangedData	13
binOverFeature	14
ChIPpeakAnno-deprecated	15
condenseMatrixByColnames	16
convert2EntrezID	17
countPatternInSeqs	18
egOrgMap	19
enrichedGO	19
ExonPlusUtr.human.GRCh37	21
findOverlappingPeaks	22
findOverlapsOfPeaks	24
findVennCounts	25
getAllPeakSequence	26
getAnnotation	27
getEnrichedGO	28
getEnrichedPATH	30
getVennCounts	32
GFF2RangedData	33
makeVennDiagram	34
myPeakList	36
Peaks.Ste12.Replicate1	37
Peaks.Ste12.Replicate2	37
Peaks.Ste12.Replicate3	38
peaksNearBDP	39
summarizePatternInPeaks	41
toGRanges	42
translatePattern	43
TSS.human.GRCh37	44
TSS.human.GRCh38	44
TSS.human.NCBI36	45
TSS.mouse.GRCm38	46
TSS.mouse.NCBIM37	46
TSS.rat.RGSC3.4	47
TSS.rat.Rnor_5.0	48

TSS.zebrafish.Zv8	48
TSS.zebrafish.Zv9	49
write2FASTA	50

Index	51
--------------	-----------

ChIPpeakAnno-package *Batch annotation of the peaks identified from either ChIP-seq or ChIP-chip experiments.*

Description

The package includes functions to retrieve the sequences around the peak, obtain enriched Gene Ontology (GO) terms, find the nearest gene, exon, miRNA or custom features such as most conserved elements and other transcription factor binding sites leveraging biomaRt, IRanges, Biostrings, BSgenome, GO.db, hypergeometric test phyper and multtest package.

Details

Package:	ChIPpeakAnno
Type:	Package
Version:	3.0.0
Date:	2014-10-24
License:	LGPL
LazyLoad:	yes

Author(s)

Lihua Julie Zhu, Jianhong Ou, Herve Pages, Claude Gazin, Nathan Lawson, Simon Lin, David Lapointe and Michael Green

Maintainer: Jianhong Ou <jianhong.ou@umassmed.edu>, Lihua Julie Zhu <julie.zhu@umassmed.edu>

References

1. Y. Benjamini and Y. Hochberg (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc. B.* Vol. 57: 289-300.
2. Y. Benjamini and D. Yekutieli (2001). The control of the false discovery rate in multiple hypothesis testing under dependency. *Annals of Statistics*. Accepted.
3. S. Durinck et al. (2005) BioMart and Bioconductor: a powerful link between biological biomarts and microarray data analysis. *Bioinformatics*, 21, 3439-3440.
4. S. Dudoit, J. P. Shaffer, and J. C. Boldrick (Submitted). Multiple hypothesis testing in microarray experiments.
5. Y. Ge, S. Dudoit, and T. P. Speed. Resampling-based multiple testing for microarray data hypothesis, Technical Report #633 of UCB Stat. <http://www.stat.berkeley.edu/~gyc>

6. Y. Hochberg (1988). A sharper Bonferroni procedure for multiple tests of significance, *Biometrika*. Vol. 75: 800-802.
7. S. Holm (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Statist.*. Vol. 6: 65-70.
8. N. L. Johnson, S. Kotz and A. W. Kemp (1992) *Univariate Discrete Distributions*, Second Edition. New York: Wiley
9. Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. *BMC Bioinformatics* 2010, 11:237doi:10.1186/1471-2105-11-237.

See Also

getAnnotation, annotatePeakInBatch, getAllPeakSequence, write2FASTA, convert2EntrezID, addAncestors, getEnrichedGO, BED2RangedData, GFF2RangedData, makeVennDiagram, findOverlappingPeaks, addGeneIDs, peaksNearBDP, summarizePatternInPeaks)

Examples

```
data(myPeakList)
data(TSS.human.NCBI36)
annotatedPeak <- annotatePeakInBatch(myPeakList[1:6],
  AnnotationData=TSS.human.NCBI36)
```

addAncestors	<i>Add GO ids of the ancestors for a given vector of GO ids</i>
--------------	---

Description

Add GO ids of the ancestors for a given vector of GO ids leveraging GO.db package

Usage

```
addAncestors(go.ids, ontology = c("bp", "cc", "mf"))
```

Arguments

go.ids	matrix with 4 columns: first column is GO IDs and 4th column is entrez IDs.
ontology	bp for biological process, cc for cellular component and mf for molecular function

Value

a vector of GO IDs containing the input GO IDs with the GO IDs of their ancestors added

Author(s)

Lihua Julie Zhu

Examples

```
go.ids = cbind(c("GO:0008150", "GO:0005576", "GO:0003674"),c("ND", "IDA", "ND"),
c("BP", "BP", "BP"), c("1", "1", "1"))
addAncestors(go.ids, ontology="bp")
```

addGeneIDs	<i>Add common IDs to annotated peaks such as gene symbol, entrez ID, ensemble gene id and refseq id.</i>
------------	--

Description

Add common IDs to annotated peaks such as gene symbol, entrez ID, ensemble gene id and refseq id leveraging organism annotation dataset! For example, org.Hs.eg.db is the dataset from orgs.Hs.eg.db package for human, while org.Mm.eg.db is the dataset from the org.Mm.eg.db package for mouse

Usage

```
addGeneIDs(annotatedPeak, orgAnn, IDs2Add=c("symbol"),
           feature_id_type="ensembl_gene_id", silence=TRUE, mart)
```

Arguments

annotatedPeak	RangedData or GRanges such as data(annotatedPeak) or a vector of feature IDs
orgAnn	organism annotation dataset such as org.Hs.eg.db
IDs2Add	a vector of annotation identifiers to be added
feature_id_type	type of ID to be annotated
silence	TRUE or FALSE. If TRUE, will not show unmapped entrez id for feature ids.
mart	mart object, see useMart of biomaRt package for details

Details

One of orgAnn and mart should be assigned.

- When orgAnn is given, parameter feature_id_type should be ensemble_gene_id, entrez_id, gene_symbol, gene_alias or refseq_id. And parameter IDs2Add can be set to any combination of identifiers such as "accnum", "ensembl", "ensemblprot", "ensembltrans", "entrez_id", "enzyme", "genename", "pfam", "pmid", "prosite", "refseq", "symbol", "unigene" and "uniprot". Some IDs are unique to a organism, such as "omim" for org.Hs.eg.db and "mgi" for org.Mm.eg.db.

Here is the definition of different IDs :

- accnum: GenBank accession numbers
- ensembl: Ensembl gene accession numbers
- ensemblprot: Ensembl protein accession numbers

- ensembltrans: Ensembl transcript accession numbers
 - entrez_id: entrez gene identifiers
 - enzyme: EC numbers
 - genename: gene name
 - pfam: Pfam identifiers
 - pmid: PubMed identifiers
 - prosite: PROSITE identifiers
 - refseq: RefSeq identifiers
 - symbol: gene abbreviations
 - unigene: UniGene cluster identifiers
 - uniprot: Uniprot accession numbers
 - omim: OMIM(Mendelian Inheritance in Man) identifiers
 - mgi: Jackson Laboratory MGI gene accession numbers
- When mart is used instead of orgAnn, for valid parameter feature_id_type and IDs2Add parameters, Please refer to [getBM](#) in bioMart package. Parameter feature_id_type should be one valid filter name listed by [listFilters\(mart\)](#) and valid attributes name listed by [listAttributes\(mart\)](#) such as `ensembl_gene_id`. And parameter IDs2Add should be one or more valid attributes name listed by [listAttributes\(mart\)](#) such as `external_gene_id`, `entrezgene`, `wiki-gene_name`, `mirbase_transcript_name`.

Value

RangedData if the input is a RangedData or dataframe with added IDs if input is a character vector.

Author(s)

Jianhong Ou, Lihua Julie Zhu

References

<http://www.bioconductor.org/packages/release/data/annotation/>

See Also

[getBM](#), [AnnotationDbi](#)

Examples

```
data(annotatedPeak)
library(org.Hs.eg.db)
addGeneIDs(annotatedPeak[1:6,],orgAnn="org.Hs.eg.db",IDs2Add=c("symbol","omim"))
addGeneIDs(annotatedPeak$feature[1:6],orgAnn="org.Hs.eg.db",IDs2Add=c("symbol","genename"))
if(interactive()){
  mart <- useMart("ENSEMBL_MART_ENSEMBL",host="www.ensembl.org",dataset="hsapiens_gene_ensembl")
  ##mart <- useMart(biomart="ensembl",dataset="hsapiens_gene_ensembl")
  addGeneIDs(annotatedPeak[1:6,],mart=mart,IDs2Add=c("hgnc_symbol","entrezgene"))
}
```

annotatedPeak	<i>Annotated Peaks</i>
---------------	------------------------

Description

TSS annotated putative STAT1-binding regions that are identified in un-stimulated cells using ChIP-seq technology (Robertson et al., 2007)

Usage

```
data(annotatedPeak)
```

Format

GRanges with slot start holding the start position of the peak, slot end holding the end position of the peak, slot names holding the id of the peak, slot strand holding the strands and slot space holding the chromosome location where the peak is located. In addition, the following variables are included.

feature id of the feature such as ensembl gene ID

insideFeature upstream: peak resides upstream of the feature; downstream: peak resides downstream of the feature; inside: peak resides inside the feature; overlapStart: peak overlaps with the start of the feature; overlapEnd: peak overlaps with the end of the feature; includeFeature: peak include the feature entirely

distancetoFeature distance to the nearest feature such as transcription start site

start_position start position of the feature such as gene

end_position end position of the feature such as the gene

Details

obtained by `data(TSS.human.GRCh37)` `data(myPeakList)` `annotatePeakInBatch (myPeakList, AnnotationData = TSS.human.GRCh37, output="b",multiple=F)`

Examples

```
data(annotatedPeak)
str(annotatedPeak)
if (interactive()) {
  y = annotatedPeak$distancetoFeature[!is.na(annotatedPeak$distancetoFeature)]
  hist(as.numeric(as.character(y)), xlab="Distance To Nearest TSS", main="", breaks=1000,
  ylim=c(0, 50), xlim=c(min(as.numeric(as.character(y)))-100,
  max(as.numeric(as.character(y)))+100))
}
```

annotatePeakInBatch *obtain the distance to the nearest TSS, miRNA, exon et al for a list of peak intervals*

Description

obtain the distance to the nearest TSS, miRNA, exon et al for a list of peak locations leveraging IRanges and biomaRt package

Usage

```
annotatePeakInBatch(myPeakList, mart, featureType = c("TSS", "miRNA", "Exon"),
  AnnotationData, output=c("nearestLocation", "overlapping", "both",
    "shortestDistance", "inside",
    "upstream&inside", "inside&downstream",
    "upstream", "downstream",
    "upstreamORdownstream"),
  multiple=c(TRUE, FALSE),
  maxgap=0L, PeakLocForDistance = c("start", "middle", "end"),
  FeatureLocForDistance = c("TSS", "middle", "start", "end", "geneEnd"),
  select=c("all", "first", "last", "arbitrary"),
  ignore.strand=TRUE)
```

Arguments

myPeakList	An object of GRanges or RangedData : See example below
mart	used if AnnotationData not supplied, a mart object, see useMart of bioMaRt package for details
featureType	used if AnnotationData not supplied, TSS, miRNA or exon
AnnotationData	annotation data obtained from getAnnotation or customized annotation of class RangedData or GRanges containing additional variable: strand (1 or + for plus strand and -1 or - for minus strand). For example, data(TSS.human.NCBI36), data(TSS.mouse.NCBIM37), data(TSS.rat.RGSC3.4) and data(TSS.zebrafish.Zv8) . If not supplied, then annotation will be obtained from biomaRt automatically using the parameters of mart and featureType
output	nearestLocation (default): will output the nearest features calculated as Peak-LocForDistance - FeatureLocForDistance; overlapping: will output overlapping features with maximum gap specified as maxgap between peak range and feature range; shortestDistance: will output nearest features; both: will output all the nearest features, in addition, will output any features that overlap the peak that is not the nearest features. upstream&inside: will output all upstream and overlapping features with maximum gap. inside&downstream: will output all downstream and overlapping features with maximum gap. upstream: will output all upstream features with maximum gap. downstream: will output all downstream features with maximum gap. upstreamORdownstream: will output all upstream features with maximum gap or downstream with maximum gap.

multiple	not applicable when output is nearest. TRUE: output multiple overlapping features for each peak. FALSE: output at most one overlapping feature for each peak. This parameter is kept for backward compatibility, please use select.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping
PeakLocForDistance	Specify the location of peak for calculating distance,i.e., middle means using middle of the peak to calculate distance to feature, start means using start of the peak to calculate the distance to feature. To be compatible with previous version, by default using start
FeatureLocForDistance	Specify the location of feature for calculating distance,i.e., middle means using middle of the feature to calculate distance of peak to feature, start means using start of the feature to calculate the distance to feature, TSS means using start of feature when feature is on plus strand and using end of feature when feature is on minus strand, geneEnd means using end of feature when feature is on plus strand and using start of feature when feature is on minus strand. To be compatible with previous version, by default using TSS
select	all may return multiple overlapping peaks, first will return the first overlapping peak, last will return the last overlapping peak and arbitrary will return one of the overlapping peaks.
ignore.strand	When set to TRUE, the strand information is ignored in the annotation.

Value

An object of [GRanges](#) or [RangedData](#) (depend on what you input) with slot start holding the start position of the peak, slot end holding the end position of the peak, slot space holding the chromosome location where the peak is located, slot rownames holding the id of the peak. In addition, the following variables are included.

feature	id of the feature such as ensembl gene ID
insideFeature	upstream: peak resides upstream of the feature; downstream: peak resides downstream of the feature; inside: peak resides inside the feature; overlapStart: peak overlaps with the start of the feature; overlapEnd: peak overlaps with the end of the feature; includeFeature: peak include the feature entirely
distancetoFeature	distance to the nearest feature such as transcription start site. By default, the distance is calculated as the distance between the start of the binding site and the TSS that is the gene start for genes located on the forward strand and the gene end for genes located on the reverse strand. The user can specify the location of peak and location of feature for calculating this
start_position	start position of the feature such as gene
end_position	end position of the feature such as the gene
strand	1 or + for positive strand and -1 or - for negative strand where the feature is located
shortestDistance	The shortest distance from either end of peak to either end the feature.

fromOverlappingOrNearest

nearest: indicates this feature's start (feature's end for features at minus strand) is closest to the peak start; Overlapping: indicates this feature overlaps with this peak although it is not the nearest feature start

Author(s)

Lihua Julie Zhu, Jianhong Ou

References

Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC Bioinformatics 2010, 11:237doi:10.1186/1471-2105-11-237

See Also

[findOverlappingPeaks](#), [makeVennDiagram](#), [addGeneIDs](#), [peaksNearBDP](#), [summarizePatternInPeaks](#)

Examples

```
#if (interactive()){
## example 1: annotate myPeakList (RangedData)
## with TSS.human.NCBI36 (RangedData)
data(myPeakList)
data(TSS.human.NCBI36)
annotatedPeak = annotatePeakInBatch(myPeakList[1:6,],
                                  AnnotationData=TSS.human.NCBI36)
annotatedPeak

## example 2: you have a list of transcription factor binding sites from
## literature and are interested in determining the extent of the overlap
## to the list of peaks from your experiment. Prior calling the function
## annotatePeakInBatch, need to represent both dataset as RangedData
## where start is the start of the binding site, end is the end of the
## binding site, names is the name of the binding site, space and strand
## are the chromosome name and strand where the binding site is located.

myexp <- GRanges(seqnames=c(6,6,6,6,5,4,4),
                 IRanges(start=c(1543200,1557200,1563000,1569800,
                               167889600,100,1000),
                       end=c(1555199,1560599,1565199,1573799,
                              167893599,200,1200),
                       names=c("p1","p2","p3","p4","p5","p6","p7")),
                 strand="+")
literature <- GRanges(seqnames=c(6,6,6,6,5,4,4),
                    IRanges(start=c(1549800,1554400,1565000,1569400,
                                   167888600,120,800),
                          end=c(1550599,1560799,1565399,1571199,
                                 167888999,140,1400),
                          names=c("f1","f2","f3","f4","f5","f6","f7")),
                    strand=rep(c("+", "-"), c(5, 2)))
```

```

annotatedPeak1 <- annotatePeakInBatch(myexp,
                                   AnnotationData=literature)
pie(table(annotatedPeak1$insideFeature))
annotatedPeak1
### use toGRanges or rtracklayer::import to convert BED or GFF format
### to GRanges before calling annotatePeakInBatch
test.bed <- data.frame(space=c("4", "6"),
                      start=c("100", "1000"),
                      end=c("200", "1100"),
                      name=c("peak1", "peak2"))
test.GR = toGRanges(test.bed)
annotatePeakInBatch(test.GR, AnnotationData = literature)
#}

```

assignChromosomeRegion

Summarizing peak distribution over exon, intron, enhancer, proximal promoter, 5 prime UTR and 3 prime UTR

Description

Summarizing peak distribution over exon, intron, enhancer, proximal promoter, 5 prime UTR and 3 prime UTR

Usage

```

assignChromosomeRegion(peaks.RD, exon, TSS, utr5, utr3,
                      proximal.promoter.cutoff=1000L, immediate.downstream.cutoff=1000L,
                      nucleotideLevel=FALSE, precedence=NULL, TxDb=NULL)

```

Arguments

peaks.RD	peaks in RangedData or GRanges: See example below
exon	exon data obtained from getAnnotation or customized annotation of class RangedData containing additional variable: strand (1 or + for plus strand and -1 or - for minus strand). Will not use anymore! use TxDb instead.
TSS	TSS data obtained from getAnnotation or customized annotation of class RangedData containing additional variable: strand (1 or + for plus strand and -1 or - for minus strand). For example, data(TSS.human.NCBI36), data(TSS.mouse.NCBIM37), data(TSS.rat.RGSC3.4) and data(TSS.zebrafish.Zv8). Will not use anymore! use TxDb instead.
utr5	5 prime UTR data obtained from getAnnotation or customized annotation of class RangedData containing additional variable: strand (1 or + for plus strand and -1 or - for minus strand). Will not use anymore! use TxDb instead.
utr3	3 prime UTR data obtained from getAnnotation or customized annotation of class RangedData containing additional variable: strand (1 or + for plus strand and -1 or - for minus strand). Will not use anymore! use TxDb instead.

proximal.promoter.cutoff	Specify the cutoff in bases to be classified as proximal promoter region. Peaks that reside within proximal.promoter.cutoff upstream from or overlap with transcription start site are classified as proximal promoters. Peaks that reside upstream over proximal.promoter.cutoff from gene start are classified as enhancers. The default is 1000 bases.
immediate.downstream.cutoff	Specify the cutoff in bases to be classified as immediate downstream. Peaks that reside within immediate.downstream.cutoff downstream of gene end but not overlap 3 prime UTR are classified as immediate downstream. Peaks that reside downstream over immediate.downstream.cutoff from gene end are classified as enhancers. The default is 1000 bases.
nucleotideLevel	NucleotideLevel (TRUE or FALSE) to allow both peak centric and nucleotide centric view. Default=FALSE
precedence	If no precedence specified, double count will be enabled, which means that if a peak overlap with both promoter and 5'UTR, then both promoter and 5'UTR will be incremented. If a precedence order is specified, for example, if promoter is specified before 5'UTR, then only promoter will be incremented for the same example. The values could be any combinations of "Promoters", "immediateDownstream", "fiveUTRs", "threeUTRs", "Exons" and "Introns", Default=NULL
TxDb	an object of TxDb

Value

jaccard	Jaccard Index
Exons	Percent of peaks reside in exon regions.
Introns	Percent of peaks reside in intron regions.
fiveUTRs	Percent of peaks reside in 5 prime UTR regions.
threeUTRs	Percent of peaks reside in 3 prime UTR regions.
Promoter	Percent of peaks reside in proximal promoter regions.
ImmediateDownstream	Percent of peaks reside in immediate downstream regions.
Enhancer.Silencer	Percent of peaks reside in enhancer/silencer regions.
queryHits	GRanges of hitted in each regions.

Author(s)

Jianhong Ou, Lihua Julie Zhu

References

Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC Bioinformatics 2010, 11:237doi:10.1186/1471-2105-11-237

See Also

annotatePeakInBatch, findOverlapsOfPeaks, getEnriched, makeVennDiagram, addGeneIDs, peaksNearBDP, summarizePattern

Examples

```

if (interactive()){
  ##Display the list of genomes available at UCSC:
  #library(rtracklayer)
  #ucscGenomes()[, "db"]
  ## Display the list of Tracks supported by makeTranscriptDbFromUCSC()
  #supportedUCSCTables()
  ##Retrieving a full transcript dataset for Human from UCSC
  ##TranscriptDb <-
  ##   makeTranscriptDbFromUCSC(genome="hg19", tablename="ensGene")
  if(require(TxDb.Hsapiens.UCSC.hg19.knownGene)){
    TxDb <- TxDb.Hsapiens.UCSC.hg19.knownGene
    exons <- exons(TxDb, columns=NULL)
    fiveUTRs <- unique(unlist(fiveUTRsByTranscript(TxDb)))
    Feature.distribution <-
      assignChromosomeRegion(exons, nucleotideLevel=TRUE, TxDb=TxDb)
    barplot(Feature.distribution$percentage)
    assignChromosomeRegion(fiveUTRs, nucleotideLevel=FALSE, TxDb=TxDb)
    data(myPeakList)
    assignChromosomeRegion(myPeakList, nucleotideLevel=TRUE,
                          precedence=c("Promoters", "immediateDownstream",
                                         "fiveUTRs", "threeUTRs",
                                         "Exons", "Introns"),
                          TxDb=TxDb)
  }
}

```

BED2RangedData	<i>convert BED format to RangedData</i>
----------------	---

Description

convert BED format to RangedData

Usage

```
BED2RangedData(data.BED, header=FALSE, ...)
```

Arguments

data.BED	BED format data frame or BED filename, please refer to http://genome.ucsc.edu/FAQ/FAQformat#format for details
header	TRUE or FALSE, default to FALSE, indicates whether data.BED file has BED header
...	any parameter need to be passed into read.delim function

Value

RangedData with slot start holding the start position of the feature, slot end holding the end position of the feature, slot names holding the id of the feature, slot space holding the chromosome location where the feature is located. In addition, the following variables are included.

strand 1 for positive strand and -1 for negative strand where the feature is located.
Default to 1 if not present in the BED formatted data frame

Note

For converting the peakList in BED format to RangedData before calling annotatePeakInBatch function

Author(s)

Lihua Julie Zhu

Examples

```
test.bed = data.frame(cbind(chrom = c("1", "2"), chromStart=c("100", "1000"),
chromEnd=c("200", "1100"), name=c("peak1", "peak2")))
test.rangedData = BED2RangedData(test.bed)
```

binOverFeature	<i>peak aggregation over bins from TSS</i>
----------------	--

Description

peak aggregation over bins from feature sites.

Usage

```
binOverFeature(..., annotationData=GRanges(),
select=c("all", "nearest"),
radius=5000L, nbins=50L,
minGeneLen=1L, aroundGene=FALSE, mbins=nbins,
featureSite=c("FeatureStart", "FeatureEnd", "bothEnd"),
PeakLocForDistance=c("all", "end", "start", "middle"),
FUN=sum, xlab, ylab, main)
```

Arguments

... objects of GRanges to be analyzed
annotationData an object of GRanges for annotation
select annotate the peaks to all features or the nearest one
radius radius of the longest distance to feature site

<code>nbins</code>	number of bins
<code>minGeneLen</code>	minimal gene length
<code>aroundGene</code>	count peaks around features or a give site of the features
<code>mbins</code>	if <code>aroundGene</code> set as TRUE, the number of bins intra-feature
<code>featureSite</code>	which site of features should be used for distance calculation
<code>PeakLocForDistance</code>	which site of peaks should be used for distance calculation
<code>FUN</code>	the function to be used for score calculation
<code>xlab</code>	titles for each x axis
<code>ylab</code>	titles for each y axis
<code>main</code>	overall titles for each plot

Value

an object of data.frame with bin values.

Author(s)

Jianhong Ou

Examples

```
bed <- system.file("extdata", "MACS_output.bed", package="ChIPpeakAnno")
gr1 <- toGRanges(bed, format="BED", header=FALSE)
data(TSS.human.GRCh37)
binOverFeature(gr1, annotationData=TSS.human.GRCh37,
               radius=5000, nbins=10, FUN=length)
```

ChIPpeakAnno-deprecated

Deprecated Functions in Package ChIPpeakAnno

Description

These functions are provided for compatibility with older versions of R only, and may be defunct as soon as the next release.

Usage

```
findOverlappingPeaks(Peaks1, Peaks2, maxgap = 0L,
                    minoverlap=1L, multiple = c(TRUE, FALSE),
                    NameOfPeaks1 = "TF1", NameOfPeaks2 = "TF2",
                    select=c("all", "first", "last", "arbitrary"),
                    annotate = 0, ignore.strand=TRUE,
                    connectedPeaks=c("min", "merge"), ...)
```

Arguments

Peaks1	RangedData: See example below.
Peaks2	RangedData: See example below.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping.
minoverlap	Non-negative integer. Intervals with an overlapping of minoverlap or more are considered to be overlapping.
multiple	TRUE or FALSE: TRUE may return multiple overlapping peaks in Peaks2 for one peak in Peaks1; FALSE will return at most one overlapping peaks in Peaks2 for one peak in Peaks1. This parameter is kept for backward compatibility, please use select.
NameOfPeaks1	Name of the Peaks1, used for generating column name.
NameOfPeaks2	Name of the Peaks2, used for generating column name.
select	all may return multiple overlapping peaks, first will return the first overlapping peak, last will return the last overlapping peak and arbitrary will return one of the overlapping peaks.
annotate	Include overlapFeature and shortestDistance in the OverlappingPeaks or not. 1 means yes and 0 means no. Default to 0.
ignore.strand	When set to TRUE, the strand information is ignored in the overlap calculations.
connectedPeaks	If multiple peaks involved in overlapping in several groups, set it to "merge" will count it as only 1, while set it to "min" will count it as the minimal involved peaks in any concered groups
...	Objects of GRanges or RangedData : See also findOverlapsOfPeaks .

Details

findOverlappingPeaks is now deprecated wrappers for [findOverlapsOfPeaks](#)

See Also

[Deprecated](#), [findOverlapsOfPeaks](#)

condenseMatrixByColnames

condense matrix by colnames

Description

condense matrix by colnames

Usage

```
condenseMatrixByColnames(mx, iname, sep=";", cnt=FALSE)
```


Arguments

mx	a matrix to be condensed
iname	the name of the column to be condensed
sep	separator for condensed values,default ;
cnt	TRUE/FALSE specifying whether adding count column or not?

Value

dataframe of condensed matrix

Author(s)

Jianhong Ou, Lihua Julie Zhu

Examples

```
a<-matrix(c(rep(rep(1:5,2),2),rep(1:10,2)),ncol=4)
colnames(a)<-c("con.1","con.2","index.1","index.2")
condenseMatrixByColnames(a,"con.1")
condenseMatrixByColnames(a,2)
```

convert2EntrezID	<i>Convert other common IDs such as ensemble gene id, gene symbol, refseq id to entrez gene ID.</i>
------------------	---

Description

Convert other common IDs such as ensemble gene id, gene symbol, refseq id to entrez gene ID leveraging organism annotation dataset! For example, org.Hs.eg.db is the dataset from orgs.Hs.eg.db package for human, while org.Mm.eg.db is the dataset from the org.Mm.eg.db package for mouse.

Usage

```
convert2EntrezID(IDs, orgAnn, ID_type="ensembl_gene_id")
```

Arguments

IDs	a vector of IDs such as ensembl gene ids
orgAnn	organism annotation dataset such as org.Hs.eg.db
ID_type	type of ID: can be ensembl_gene_id, gene_symbol or refseq_id

Value

vector of entrez ids

Author(s)

Lihua Julie Zhu

Examples

```
ensemblIDs = c("ENSG00000115956", "ENSG00000071082", "ENSG00000071054",
              "ENSG00000115594", "ENSG00000115594", "ENSG00000115598", "ENSG00000170417")
library(org.Hs.eg.db)
entrezIDs = convert2EntrezID(IDs=ensemblIDs, orgAnn="org.Hs.eg.db",
                             ID_type="ensembl_gene_id")
```

countPatternInSeqs	<i>Output total number of patterns found in the input sequences</i>
--------------------	---

Description

Output total number of patterns found in the input sequences

Usage

```
countPatternInSeqs(pattern, sequences)
```

Arguments

pattern	DNAstringSet object
sequences	a vector of sequences

Value

Total number of occurrence of the pattern in the sequences

Author(s)

Lihua Julie Zhu

See Also

summarizePatternInPeaks, translatePattern

Examples

```
filepath = system.file("extdata", "examplePattern.fa", package="ChIPpeakAnno")
dict = readDNAStringSet(filepath = filepath, format="fasta", use.names=TRUE)
sequences = c("ACTGGGGGGGCTGGGCCCAAAAT",
             "AAAAAACCCCTTTTGCCATCCCGGGACGGGCCCAT",
             "ATCGAAAATTTCC")
countPatternInSeqs(pattern=dict[1], sequences=sequences)
countPatternInSeqs(pattern=dict[2], sequences=sequences)
pattern = DNAStringSet("ATNGMAA")
countPatternInSeqs(pattern=pattern, sequences=sequences)
```

egOrgMap	<i>map organism annotation dataset to specie name or reverse.</i>
----------	---

Description

Give a specie name and return the organism annotation dataset name or give a organism annotation dataset name then return the specie name.

Usage

```
egOrgMap(name)
```

Arguments

name	organism annotation dataset or the specie name.
------	---

Value

a object of character

Author(s)

Jianhong Ou

Examples

```
egOrgMap("org.Hs.eg.db")  
egOrgMap("Mus musculus")
```

enrichedGO	<i>Enriched Gene Ontology terms used as example</i>
------------	---

Description

Enriched Gene Ontology terms used as example

Usage

```
data(enrichedGO)
```

Format

A list of 3 variables.

bp enriched biological process with 9 variables

go.id:GO biological process id
go.term:GO biological process term
go.Definition:GO biological process description
Ontology: Ontology branch, i.e. BP for biological process
count.InDataset: count of this GO term in this dataset
count.InGenome: count of this GO term in the genome
pvalue: pvalue from the hypergeometric test
totaltermInDataset: count of all GO terms in this dataset
totaltermInGenome: count of all GO terms in the genome

mf enriched molecular function with the following 9 variables

go.id:GO molecular function id
go.term:GO molecular function term
go.Definition:GO molecular function description
Ontology: Ontology branch, i.e. MF for molecular function
count.InDataset: count of this GO term in this dataset
count.InGenome: count of this GO term in the genome
pvalue: pvalue from the hypergeometric test
totaltermInDataset: count of all GO terms in this dataset
totaltermInGenome: count of all GO terms in the genome

cc enriched cellular component the following 9 variables

go.id:GO cellular component id
go.term:GO cellular component term
go.Definition:GO cellular component description
Ontology: Ontology type, i.e. CC for cellular component
count.InDataset: count of this GO term in this dataset
count.InGenome: count of this GO term in the genome
pvalue: pvalue from the hypergeometric test
totaltermInDataset: count of all GO terms in this dataset
totaltermInGenome: count of all GO terms in the genome

Author(s)

Lihua Julie Zhu

Examples

```
data(enrichedGO)
dim(enrichedGO$mf)
dim(enrichedGO$cc)
dim(enrichedGO$bp)
```

ExonPlusUtr.human.GRCh37

Gene model with exon, 5' UTR and 3' UTR information for human sapiens (GRCh37) obtained from biomaRt

Description

Gene model with exon, 5' UTR and 3' UTR information for human sapiens (GRCh37) obtained from biomaRt

Usage

```
data(ExonPlusUtr.human.GRCh37)
```

Format

RangedData with slot start holding the start position of the exon, slot end holding the end position of the exon, slot rownames holding ensembl transcript id and slot space holding the chromosome location where the gene is located. In addition, the following variables are included.

strand 1 for positive strand and -1 for negative strand

description description of the transcript

ensembl_gene_id gene id

utr5start 5' UTR start

utr5end 5' UTR end

utr3start 3' UTR start

utr3end 3' UTR end

Details

used in the examples Annotation data obtained by: `mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")` `ExonPlusUtr.human.GRCh37 = getAnnotation(mart=human, featureType="ExonPlusUtr")`

Examples

```
data(ExonPlusUtr.human.GRCh37)
slotNames(ExonPlusUtr.human.GRCh37)
```

findOverlappingPeaks *Find the overlapping peaks for two peak ranges.*

Description

Find the overlapping peaks for two input peak ranges.

Usage

```
findOverlappingPeaks(Peaks1, Peaks2, maxgap = 0L,
  minoverlap=1L, multiple = c(TRUE, FALSE),
  NameOfPeaks1 = "TF1", NameOfPeaks2 = "TF2",
  select=c("all", "first", "last", "arbitrary"), annotate = 0,
  ignore.strand=TRUE,
  connectedPeaks=c("min", "merge"), ...)
```

Arguments

Peaks1	RangedData: See example below.
Peaks2	RangedData: See example below.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping.
minoverlap	Non-negative integer. Intervals with an overlapping of minoverlap or more are considered to be overlapping.
multiple	TRUE or FALSE: TRUE may return multiple overlapping peaks in Peaks2 for one peak in Peaks1; FALSE will return at most one overlapping peaks in Peaks2 for one peak in Peaks1. This parameter is kept for backward compatibility, please use select.
NameOfPeaks1	Name of the Peaks1, used for generating column name.
NameOfPeaks2	Name of the Peaks2, used for generating column name.
select	all may return multiple overlapping peaks, first will return the first overlapping peak, last will return the last overlapping peak and arbitrary will return one of the overlapping peaks.
annotate	Include overlapFeature and shortestDistance in the OverlappingPeaks or not. 1 means yes and 0 means no. Default to 0.
ignore.strand	When set to TRUE, the strand information is ignored in the overlap calculations.
connectedPeaks	If multiple peaks involved in overlapping in several groups, set it to "merge" will count it as only 1, while set it to "min" will count it as the minimal involved peaks in any concentered groups
...	Objects of GRanges or RangedData : See also findOverlapsOfPeaks .

Details

Efficiently perform overlap queries with an interval tree implemented in IRanges.

Value

OverlappingPeaks

a data frame consists of input peaks information with added information: overlapFeature (upstream: peak1 resides upstream of the peak2; downstream: peak1 resides downstream of the peak2; inside: peak1 resides inside the peak2 entirely; overlapStart: peak1 overlaps with the start of the peak2; overlapEnd: peak1 overlaps with the end of the peak2; includeFeature: peak1 include the peak2 entirely) and shortestDistance (shortest distance between the overlapping peaks)

MergedPeaks RangedData contains merged overlapping peaks

Author(s)

Lihua Julie Zhu

References

1.Interval tree algorithm from: Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms, second edition, MIT Press and McGraw-Hill. ISBN 0-262-53196-8 2.Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC Bioinformatics 2010, 11:237doi:10.1186/1471-2105-11-237

See Also

annotatePeakInBatch, makeVennDiagram

Examples

```
if (interactive())
{
peaks1 = RangedData(IRanges(start=c(1543200,1557200,1563000,1569800,167889600),
end=c(1555199,1560599,1565199,1573799,167893599),names=c("p1","p2","p3","p4","p5")),
strand=as.integer(1),space=c(6,6,6,6,5))
peaks2 = RangedData(IRanges(start=c(1549800,1554400,1565000,1569400,167888600),
end=c(1550599,1560799,1565399,1571199,167888999),names=c("f1","f2","f3","f4","f5")),
strand=as.integer(1),space=c(6,6,6,6,5))
t1 =findOverlappingPeaks(peaks1, peaks2, maxgap=1000,
NameOfPeaks1="TF1", NameOfPeaks2="TF2", select="all", annotate=1)
r = t1$OverlappingPeaks
pie(table(r$overlapFeature))
as.data.frame(t1$MergedPeaks)
}
```

findOverlapsOfPeaks *Find the overlapping peaks for two or more peak ranges.*

Description

Find the overlapping peaks for two or more (less than five) peak ranges.

Usage

```
findOverlapsOfPeaks(..., maxgap=0L, minoverlap=1L,
                    ignore.strand=TRUE, connectedPeaks=c("min", "merge", "keepAll"))
```

Arguments

...	Objects of GRanges or RangedData : See example below.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping.
minoverlap	Non-negative integer. Intervals with an overlapping of minoverlap or more are considered to be overlapping.
ignore.strand	When set to TRUE, the strand information is ignored in the overlap calculations.
connectedPeaks	If multiple peaks involved in overlapping in several groups, set it to "merge" will count it as only 1, while set it to "min" will count it as the minimal involved peaks in any concered groups

Details

Efficiently perform overlap queries with an interval tree implemented in [GRanges](#).

Value

return value is An object of overlappingPeaks.

venn_cnt	an object of VennCounts
peaklist	a list consists of all overlapping peaks or unique peaks

Author(s)

Jianhong Ou

References

- Interval tree algorithm from: Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms, second edition, MIT Press and McGraw-Hill. ISBN 0-262-53196-8
- Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC Bioinformatics 2010, 11:237doi:10.1186/1471-2105-11-237

See Also

[annotatePeakInBatch](#), [makeVennDiagram](#), [getVennCounts](#), [findOverlappingPeaks](#)

Examples

```
peaks1 <- GRanges(seqnames=c(6,6,6,6,5),
                  IRanges(start=c(1543200,1557200,1563000,1569800,167889600),
                          end=c(1555199,1560599,1565199,1573799,167893599),
                          names=c("p1","p2","p3","p4","p5")),
                  strand="+")
peaks2 <- GRanges(seqnames=c(6,6,6,6,5),
                  IRanges(start=c(1549800,1554400,1565000,1569400,167888600),
                          end=c(1550599,1560799,1565399,1571199,167888999),
                          names=c("f1","f2","f3","f4","f5")),
                  strand="+")
t1 <- findOverlapsOfPeaks(peaks1, peaks2, maxgap=1000)
makeVennDiagram(t1)
t1$venn_cnt
t1$peaklist
```

findVennCounts	<i>Obtain Venn Counts for Venn Diagram, internal function for makeVennDiagram</i>
----------------	---

Description

Obtain Venn Counts for two peak ranges using chromosome ranges or feature field, internal function for makeVennDiagram

Usage

```
findVennCounts(Peaks, NameOfPeaks, maxgap = 0L, minoverlap = 1L,
               totalTest, useFeature=FALSE)
```

Arguments

Peaks	RangedDataList: See example below.
NameOfPeaks	Character vector to specify the name of Peaks, e.g., c("TF1", "TF2"), this will be used as label in the Venn Diagram.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping.
minoverlap	Non-negative integer. Intervals with an overlapping of minoverlap or more are considered to be overlapping.
totalTest	Numeric value to specify the total number of tests performed to obtain the list of peaks.
useFeature	TRUE or FALSE, default FALSE, true means using feature field in the Ranged-Data for calculating overlap, false means using chromosome range for calculating overlap.

Value

p.value	hypergeometric testing result
vennCounts	vennCounts objects containing counts for Venn Diagram generation, see details in limma package vennCounts

Note

```
if (interactive())
peaks1 = RangedData(IRanges(start = c(967654, 2010897, 2496704), end = c(967754, 2010997,
2496804), names = c("Site1", "Site2", "Site3")), space = c("1", "2", "3"), strand=as.integer(1), fea-
ture=c("a","b", "c")) peaks2 = RangedData(IRanges(start = c(967659, 2010898, 2496700, 3075866,
3123260), end = c(967869, 2011108, 2496920, 3076166, 3123470), names = c("t1", "t2", "t3", "t4",
"t5")), space = c("1", "2", "3", "1", "2"), strand = c(1, 1, -1,-1,1), feature=c("a","c","d","e", "a"))
findVennCounts(RangedDataList(peaks1,peaks2), NameOfPeaks=c("TF1", "TF2"), maxgap=0,totalTest=100,
useFeature=FALSE) findVennCounts(RangedDataList(peaks1,peaks2), NameOfPeaks=c("TF1", "TF2"),
maxgap=0,totalTest=100, useFeature=TRUE) 
```

Author(s)

Lihua Julie Zhu

See Also

makeVennDiagram

getAllPeakSequence	<i>Obtain genomic sequences around the peaks</i>
--------------------	--

Description

Obtain genomic sequences around the peaks leveraging BSgenome and biomaRt package

Usage

```
getAllPeakSequence(myPeakList, upstream = 200L, downstream = upstream,
genome, AnnotationData)
```

Arguments

myPeakList	An object of GRanges or RangedData : See example below
upstream	upstream offset from the peak start, e.g., 200
downstream	downstream offset from the peak end, e.g., 200
genome	BSgenome object or mart object. Please refer to available.genomes in BSgenome package and useMart in bioMaRt package for details
AnnotationData	RangedData used if mart object is parsed in which can be obtained from getAn-notation with featureType="TSS". For example, data(TSS.human.NCBI36), data(TSS.mouse.NCBIM37), data(GO.rat.RGSC3.4) and data(TSS.zebrafish.Zv8). If not supplied, then anno-tation will be obtained from biomaRt automatically using the mart object

Value

[GRanges](#) or [RangedData](#) with slot start holding the start position of the peak, slot end holding the end position of the peak, slot rownames holding the id of the peak and slot space holding the chromosome location where the peak is located. In addition, the following variables are included.

upstream	upstream offset from the peak start
downstream	downstream offset from the peak end
sequence	the sequence obtained

Author(s)

Lihua Julie Zhu, Jianhong Ou

References

Durinck S. et al. (2005) BioMart and Bioconductor: a powerful link between biological biomarts and microarray data analysis. *Bioinformatics*, 21, 3439-3440.

Examples

```
#### use Annotation data from BSgenome
peaks <- GRanges(seqnames=c("NC_008253", "NC_010468"),
                 IRanges(start=c(100, 500), end=c(300, 600),
                         names=c("peak1", "peak2")))
library(BSgenome.Ecoli.NCBI.20080805)
seq <- getAllPeakSequence(peaks, upstream=20, downstream=20, genome=Ecoli)
write2FASTA(seq, file="test.fa")
```

getAnnotation

Obtain the TSS, exon or miRNA annotation for the specified species

Description

Obtain the TSS, exon or miRNA annotation for the specified species using biomaRt package

Usage

```
getAnnotation(mart,
              featureType=c("TSS", "miRNA", "Exon", "5utr", "3utr", "ExonPlusUtr", "transcript"),
              output=c("RangedData", "GRanges"))
```

Arguments

mart	mart object, see useMart of bioMaRt package for details
featureType	TSS, miRNA, Exon, 5'UTR, 3'UTR, transcript or Exon plus UTR
output	the class of output data, could be GRanges or RangedData

Value

[GRanges](#) or [RangedData](#) with slot start holding the start position of the feature, slot end holding the end position of the feature, slot names holding the id of the feature, slot space holding the chromosome location where the feature is located. In addition, the following variables are included.

strand	1 for positive strand and -1 for negative strand where the feature is located
description	description of the feature such as gene

Note

For featureType of TSS, start is the transcription start site if strand is 1 (plus strand), otherwise, end is the transcription start site

Author(s)

Lihua Julie Zhu, Jianhong Ou

References

Durinck S. et al. (2005) BioMart and Bioconductor: a powerful link between biological biomarts and microarray data analysis. *Bioinformatics*, 21, 3439-3440.

Examples

```
if (interactive())
{
  mart<-useMart(biomart="ensembl",dataset="hsapiens_gene_ensembl")
  Annotation = getAnnotation(mart, featureType="TSS")
}
```

getEnrichedGO

Obtain enriched gene ontology (GO) terms that near the peaks

Description

Obtain enriched gene ontology (GO) terms that are near the peaks using GO.db package and GO gene mapping package such as org.Hs.db.eg to obtain the GO annotation and using hypergeometric test (phyper) and multtest package for adjusting p-values

Usage

```
getEnrichedGO(annotatedPeak, orgAnn, feature_id_type="ensembl_gene_id",
maxP=0.01, multiAdj=FALSE, minGOterm=10, multiAdjMethod="")
```

Arguments

annotatedPeak	RangedData or GRanges such as data(annotatedPeak) or a vector of feature IDs
orgAnn	organism annotation package such as org.Hs.eg.db for human and org.Mm.eg.db for mouse, org.Dm.eg.db for fly, org.Rn.eg.db for rat, org.Sc.eg.db for yeast and org.Dr.eg.db for zebrafish
feature_id_type	the feature type in annotatedPeakRanges such as ensembl_gene_id, refseq_id, gene_symbol or entrez_id
maxP	maximum p-value to be considered to be significant
multiAdj	Whether apply multiple hypothesis testing adjustment, TRUE or FALSE
minGOterm	minimum count in a genome for a GO term to be included
multiAdjMethod	multiple testing procedures, for details, see mt.rawp2adjp in multtest package

Value

A list of 3

bp	<p>enriched biological process with the following 9 variables</p> <p>go.id:GO biological process id</p> <p>go.term:GO biological process term</p> <p>go.Definition:GO biological process description</p> <p>Ontology: Ontology branch, i.e. BP for biological process</p> <p>count.InDataset: count of this GO term in this dataset</p> <p>count.InGenome: count of this GO term in the genome</p> <p>pvalue: pvalue from the hypergeometric test</p> <p>totaltermInDataset: count of all GO terms in this dataset</p> <p>totaltermInGenome: count of all GO terms in the genome</p>
mf	<p>enriched molecular function with the following 9 variables</p> <p>go.id:GO molecular function id</p> <p>go.term:GO molecular function term</p> <p>go.Definition:GO molecular function description</p> <p>Ontology: Ontology branch, i.e. MF for molecular function</p> <p>count.InDataset: count of this GO term in this dataset</p> <p>count.InGenome: count of this GO term in the genome</p> <p>pvalue: pvalue from the hypergeometric test</p> <p>totaltermInDataset: count of all GO terms in this dataset</p> <p>totaltermInGenome: count of all GO terms in the genome</p>
cc	<p>enriched cellular component the following 9 variables</p> <p>go.id:GO cellular component id</p> <p>go.term:GO cellular component term</p> <p>go.Definition:GO cellular component description</p> <p>Ontology: Ontology type, i.e. CC for cellular component</p> <p>count.InDataset: count of this GO term in this dataset</p>

count.InGenome: count of this GO term in the genome
 pvalue: pvalue from the hypergeometric test
 totaltermInDataset: count of all GO terms in this dataset
 totaltermInGenome: count of all GO terms in the genome

Author(s)

Lihua Julie Zhu

References

Johnson, N. L., Kotz, S., and Kemp, A. W. (1992) Univariate Discrete Distributions, Second Edition. New York: Wiley

See Also

phyper, hyperGtest

Examples

```
data(enrichedGO)
enrichedGO$mf[1:10,]
enrichedGO$bp[1:10,]
enrichedGO$cc
if (interactive()) {
  data(annotatedPeak)
  library(org.Hs.eg.db)
  enriched.GO = getEnrichedGO(annotatedPeak[1:6,], orgAnn="org.Hs.eg.db", maxP=0.01,
    multiAdj=FALSE, minGOterm=10, multiAdjMethod="")
  dim(enriched.GO$mf)
  colnames(enriched.GO$mf)
  dim(enriched.GO$bp)
  enriched.GO$cc
}
```

getEnrichedPATH

Obtain enriched PATH that near the peaks

Description

Obtain enriched PATH that are near the peaks using path package such as reactome.db and path mapping package such as org.Hs.db.eg to obtain the path annotation and using hypergeometric test (phyper) and multtest package for adjusting p-values

Usage

```
getEnrichedPATH(annotatedPeak, orgAnn, pathAnn, feature_id_type="ensembl_gene_id",
  maxP=0.01, minPATHterm=10, multiAdjMethod=NULL)
```

Arguments

annotatedPeak	RangedData or GRanges such as data(annotatedPeak) or a vector of feature IDs
orgAnn	organism annotation package such as org.Hs.eg.db for human and org.Mm.eg.db for mouse, org.Dm.eg.db for fly, org.Rn.eg.db for rat, org.Sc.eg.db for yeast and org.Dr.eg.db for zebrafish
pathAnn	pathway annotation package such as KEGG.db, reactome.db
feature_id_type	the feature type in annotatedPeakRanges such as ensembl_gene_id, refseq_id, gene_symbol or entrez_id
maxP	maximum p-value to be considered to be significant
minPATHterm	minimum count in a genome for a path to be included
multiAdjMethod	multiple testing procedures, for details, see mt.rawp2adjp in multtest package

Value

A dataframe of enriched path with the following variables.

path.id	KEGG PATH ID
EntrezID	Entrez ID
count.InDataset	count of this PATH in this dataset
count.InGenome	count of this PATH in the genome
pvalue	pvalue from the hypergeometric test
totaltermInDataset	count of all PATH in this dataset
totaltermInGenome	count of all PATH in the genome
PATH	PATH name

Author(s)

Jianhong Ou

References

Johnson, N. L., Kotz, S., and Kemp, A. W. (1992) Univariate Discrete Distributions, Second Edition. New York: Wiley

See Also

phyper, hyperGtest

Examples

```

if (interactive()) {
  data(annotatedPeak)
  library(org.Hs.eg.db)
  library(reactome.db)
  enriched.PATH = getEnrichedPATH(annotatedPeak, orgAnn="org.Hs.eg.db",
    pathAnn="reactome.db", maxP=0.01,
    minPATHterm=10, multiAdjMethod=NULL)
  head(enriched.PATH)
}

```

getVennCounts	<i>Obtain Venn Counts for Venn Diagram, internal function for makeVennDiagram</i>
---------------	---

Description

Obtain Venn Counts for peak ranges using chromosome ranges or feature field, internal function for makeVennDiagram

Usage

```

getVennCounts(..., maxgap = 0L, minoverlap=1L, by=c("region", "feature", "base"),
  ignore.strand=TRUE, connectedPeaks=c("min", "merge", "keepAll"))

```

Arguments

...	Objects of GRanges or RangedData : See example below.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping.
minoverlap	Non-negative integer. Intervals with an overlapping of minoverlap or more are considered to be overlapping.
by	region, feature or base, default region. feature means using feature field in the RangedData or GRanges for calculating overlap, region means using chromosome range for calculating overlap, and base means using calculating overlap in nucleotide level.
ignore.strand	When set to TRUE, the strand information is ignored in the overlap calculations.
connectedPeaks	If multiple peaks involved in overlapping in several groups, set it to "merge" will count it as only 1, while set it to "min" will count it as the minimal involved peaks in any concered groups

Value

vennCounts	vennCounts objects containing counts for Venn Diagram generation, see details in limma package vennCounts
------------	---

Author(s)

Jianhong Ou

See Also[makeVennDiagram](#), [findOverlappingPeaks](#)**Examples**

```

if(interactive()){
peaks1 = RangedData(IRanges(start = c(967654, 2010897, 2496704),
                             end = c(967754, 2010997, 2496804),
                             names = c("Site1", "Site2", "Site3")),
                  space = c("1", "2", "3"),
                  strand=as.integer(1),
                  feature=c("a","b", "c"))
peaks2 = RangedData(IRanges(start=c(967659, 2010898, 2496700, 3075866, 3123260),
                             end=c(967869, 2011108, 2496920, 3076166, 3123470),
                             names = c("t1", "t2", "t3", "t4", "t5")),
                  space = c("1", "2", "3", "1", "2"),
                  strand = c(1, 1, -1,-1,1),
                  feature=c("a","c","d","e", "a"))
getVennCounts(peaks1,peaks2, maxgap=0)
getVennCounts(peaks1,peaks2, maxgap=0, by="feature")
getVennCounts(peaks1, peaks2, maxgap=0, by="base")
}

```

GFF2RangedData

*convert GFF format to RangedData***Description**

convert GFF format to RangedData

Usage

GFF2RangedData(data.GFF,header=FALSE, ...)

Arguments

data.GFF	GFF format data frame or GFF file name, please refer to http://genome.ucsc.edu/FAQ/FAQformat#format3 for details
header	TRUE or FALSE, default to FALSE, indicates whether data.GFF file has GFF header
...	any parameter need to be passed into read.delim function

Value

RangedData with slot start holding the start position of the feature, slot end holding the end position of the feature, slot names holding the id of the feature, slot space holding the chromosome location where the feature is located. In addition, the following variables are included.

strand 1 for positive strand and -1 for negative strand where the feature is located.

Note

For converting the peakList in GFF format to RangedData before calling annotatePeakInBatch function

Author(s)

Lihua Julie Zhu

Examples

```
test.GFF = data.frame(cbind(seqname = c("chr1", "chr2"), source=rep("Macs", 2),
feature=rep("peak", 2), start=c("100", "1000"), end=c("200", "1100"), score=c(60, 26),
strand=c(1, -1), frame=c(".", 2), group=c("peak1", "peak2")))
test.rangedData = GFF2RangedData(test.GFF)
```

makeVennDiagram	<i>Make Venn Diagram from two peak ranges</i>
-----------------	---

Description

Make Venn Diagram from two peak ranges and also calculate p-value for determining whether two peak ranges overlap significantly.

Usage

```
makeVennDiagram(Peaks, NameOfPeaks, maxgap=0L, minoverlap=1L, totalTest,
by=c("region", "feature", "base"), ignore.strand=TRUE,
connectedPeaks=c("min", "merge", "keepAll"), ...)
```

Arguments

Peaks	A list of GRanges or RangedData : See example below.
NameOfPeaks	Character vector to specify the name of Peaks, e.g., c("TF1", "TF2"), this will be used as label in the Venn Diagram.
maxgap	Non-negative integer. Intervals with a separation of maxgap or less are considered to be overlapping.
minoverlap	Non-negative integer. Intervals with an overlapping of minoverlap or more are considered to be overlapping.

totalTest	Numeric value to specify the total number of tests performed to obtain the list of peaks. It should be much larger than the number of peaks in the largest peak set.
by	region, feature or base, default region. feature means using feature field in the RangedData or GRanges for calculating overlap, region means using chromosome range for calculating overlap, and base means using calculating overlap in nucleotide level.
ignore.strand	When set to TRUE, the strand information is ignored in the overlap calculations.
connectedPeaks	If multiple peaks involved in overlapping in several groups, set it to "merge" will count it as only 1, while set it to "min" will count it as the minimal involved peaks in any concered groups
...	Additional arguments to be passed to venn.diagram

Details

For customized graph options, please see `venn.diagram` in `VennDiagram` package.

Value

In addition to a Venn Diagram produced, `p.value` is obtained from hypergeometric test for determining whether the two peak ranges or features overlap significantly.

Author(s)

Lihua Julie Zhu, Jianhong Ou

See Also

[findOverlappingPeaks](#), [venn.diagram](#)

Examples

```
if (interactive()){
  peaks1 <- GRanges(seqnames=c("1", "2", "3"),
                    IRanges(start=c(967654, 2010897, 2496704),
                             end=c(967754, 2010997, 2496804),
                             names=c("Site1", "Site2", "Site3")),
                    strand="+",
                    feature=c("a", "b", "f"))
  peaks2 = RangedData(seqnames=c("1", "2", "3", "1", "2"),
                      IRanges(start = c(967659, 2010898, 2496700,
                                         3075866, 3123260),
                               end = c(967869, 2011108, 2496920,
                                         3076166, 3123470),
                               names = c("t1", "t2", "t3", "t4", "t5")),
                      strand = c("+", "+", "-", "-", "+"),
                      feature=c("a", "b", "c", "d", "a"))
  makeVennDiagram(list(peaks1, peaks2), NameOfPeaks=c("TF1", "TF2"),
                  totalTest=100, scaled=FALSE, euler.d=FALSE)
```

```
makeVennDiagram(list(peaks1, peaks2), NameOfPeaks=c("TF1", "TF2"),
                  totalTest=100)

##### 4-way diagram using annotated feature instead of chromosome ranges

makeVennDiagram(list(peaks1, peaks2, peaks1, peaks2),
                  NameOfPeaks=c("TF1", "TF2", "TF3", "TF4"),
                  totalTest=100, by="feature",
                  main = "Venn Diagram for 4 peak lists",
                  fill=c(1,2,3,4))
}
```

myPeakList

ChIP-seq peak dataset

Description

the putative STAT1-binding regions identified in un-stimulated cells using ChIP-seq technology (Robertson et al., 2007)

Usage

```
data(myPeakList)
```

Format

RangedData with slot rownames containing the ID of peak as character, slot start containing the start position of the peak, slot end containing the end position of the peak and space containing the chromosome where the peak is located.

Source

Robertson G, Hirst M, Bainbridge M, Bilenky M, Zhao Y, et al. (2007) Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. Nat Methods 4:651-7

Examples

```
data(myPeakList)
slotNames(myPeakList)
```

Peaks.Ste12.Replicate1

Ste12-binding sites from biological replicate 1 in yeast (see reference)

Description

Ste12-binding sites from biological replicate 1 in yeast (see reference)

Usage

```
data(Peaks.Ste12.Replicate1)
```

Format

RangedData with slot rownames containing the ID of peak as character, slot start containing the start position of the peak, slot end containing the end position of the peak and space containing the chromosome where the peak is located.

References

Philippe Lefrançois, Ghia M Euskirchen, Raymond K Auerbach, Joel Rozowsky, Theodore Gibson, Christopher M Yellman, Mark Gerstein and Michael Snyder (2009) Efficient yeast ChIP-Seq using multiplex short-read DNA sequencing BMC Genomics 10:37

Examples

```
data(Peaks.Ste12.Replicate1)
str(Peaks.Ste12.Replicate1)
```

Peaks.Ste12.Replicate2

Ste12-binding sites from biological replicate 2 in yeast (see reference)

Description

Ste12-binding sites from biological replicate 2 in yeast (see reference)

Usage

```
data(Peaks.Ste12.Replicate2)
```

Format

RangedData with slot rownames containing the ID of peak as character, slot start containing the start position of the peak, slot end containing the end position of the peak and space containing the chromosome where the peak is located.

Source

<http://www.biomedcentral.com/1471-2164/10/37>

References

Philippe Lefrançois, Ghia M Euskirchen, Raymond K Auerbach, Joel Rozowsky, Theodore Gibson, Christopher M Yellman, Mark Gerstein and Michael Snyder (2009) Efficient yeast ChIP-Seq using multiplex short-read DNA sequencing BMC Genomics 10:37doi:10.1186/1471-2164-10-37

Examples

```
data(Peaks.Ste12.Replicate2)
str(Peaks.Ste12.Replicate2)
```

Peaks.Ste12.Replicate3

Ste12-binding sites from biological replicate 3 in yeast (see reference)

Description

Ste12-binding sites from biological replicate 3 in yeast (see reference)

Usage

```
data(Peaks.Ste12.Replicate3)
```

Format

RangedData with slot rownames containing the ID of peak as character, slot start containing the start position of the peak, slot end containing the end position of the peak and space containing the chromosome where the peak is located.

Source

<http://www.biomedcentral.com/1471-2164/10/37>

References

Philippe Lefrançois, Ghia M Euskirchen, Raymond K Auerbach, Joel Rozowsky, Theodore Gibson, Christopher M Yellman, Mark Gerstein and Michael Snyder (2009) Efficient yeast ChIP-Seq using multiplex short-read DNA sequencing BMC Genomics 10:37doi:10.1186/1471-2164-10-37

Examples

```
data(Peaks.Ste12.Replicate3)
str(Peaks.Ste12.Replicate3)
```

peaksNearBDP *obtain the peaks near bi-directional promoters*

Description

Obtain the peaks near bi-directional promoters. Also output percent of peaks near bi-directional promoters.

Usage

```
peaksNearBDP(myPeakList, mart, AnnotationData, MaxDistance=5000,
             PeakLocForDistance = c("start", "middle", "end"),
             FeatureLocForDistance = c("TSS", "middle", "start", "end", "geneEnd"))
```

Arguments

myPeakList	GRanges or RangedData : See example below
mart	used if AnnotationData not supplied, a mart object, see useMart of bioMaRt package for details
AnnotationData	annotation data obtained from getAnnotation or customized annotation of class GRanges or RangedData containing additional variable: strand (1 or + for plus strand and -1 or - for minus strand). For example, data(TSS.human.NCBI36), data(TSS.mouse.NCBIM37), data(TSS.rat.RGSC3.4) and data(TSS.zebrafish.Zv8) . If not supplied, then annotation will be obtained from biomaRt automatically using the parameters of mart and featureType TSS
MaxDistance	Specify the maximum gap allowed between the peak and nearest gene
PeakLocForDistance	Specify the location of peak for calculating distance, i.e., middle means using middle of the peak to calculate distance to feature, start means using start of the peak to calculate the distance to feature. To be compatible with previous version, by default using start
FeatureLocForDistance	Specify the location of feature for calculating distance, i.e., middle means using middle of the feature to calculate distance of peak to feature, start means using start of the feature to calculate the distance to feature, TSS means using start of feature when feature is on plus strand and using end of feature when feature is on minus strand, geneEnd means using end of feature when feature is on plus strand and using start of feature when feature is on minus strand. To be compatible with previous version, by default using TSS

Value

A list of 4

peaksWithBDP annotated Peaks containing bi-directional promoters.
 RangedData with slot start holding the start position of the peak, slot end holding the end position of the peak, slot space holding the chromosome location where the peak is located, slot rownames holding the id of the peak. In addition, the following variables are included.
 feature: id of the feature such as ensembl gene ID
 insideFeature: upstream: peak resides upstream of the feature; downstream: peak resides downstream of the feature; inside: peak resides inside the feature; overlapStart: peak overlaps with the start of the feature; overlapEnd: peak overlaps with the end of the feature; includeFeature: peak include the feature entirely.
 distancetoFeature: distance to the nearest feature such as transcription start site. By default, the distance is calculated as the distance between the start of the binding site and the TSS that is the gene start for genes located on the forward strand and the gene end for genes located on the reverse strand. The user can specify the location of peak and location of feature for calculating this
 start_position: start position of the feature such as gene
 end_position: end position of the feature such as the gene
 strand: 1 or + for positive strand and -1 or - for negative strand where the feature is located
 shortestDistance: The shortest distance from either end of peak to either end the feature
 fromOverlappingOrNearest: NearestStart: indicates this PeakLocForDistance is closest to the FeatureLocForDistance

percentPeaksWithBDP The percent of input peaks containing bi-directional promoters

n.peaks The total number of input peaks

n.peaksWithBDP The # of input peaks containing bi-directional promoters

Author(s)

Lihua Julie Zhu, Jianhong Ou

References

Zhu L.J. et al. (2010) ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data. BMC Bioinformatics 2010, 11:237doi:10.1186/1471-2105-11-237

See Also

annotatePeakInBatch, findOverlappingPeaks, makeVennDiagram

Examples

```
if (interactive())
{
data(myPeakList)
```



```

data(TSS.human.NCBI36)
annotatedBDP = peaksNearBDP(myPeakList[1:6,], AnnotationData=TSS.human.NCBI36,
MaxDistance=5000,PeakLocForDistance = "middle",
FeatureLocForDistance = "TSS")
c(annotatedBDP$percentPeaksWithBDP, annotatedBDP$n.peaks, annotatedBDP$n.peaksWithBDP)
}

```

summarizePatternInPeaks

Output a summary of the occurrence of each pattern in the sequences.

Description

Output a summary of the occurrence of each pattern in the sequences.

Usage

```

summarizePatternInPeaks(patternFilePath, format = "fasta", skip=0L,
                        BSgenomeName, peaks, outfile, append = FALSE)

```

Arguments

patternFilePath	A character vector containing the path to the file to read the patterns from.
format	Either "fasta" (the default) or "fastq"
skip	Single non-negative integer. The number of records of the pattern file to skip before beginning to read in records.
BSgenomeName	BSgenome object. Please refer to available.genomes in BSgenome package for details
peaks	GRanges or RangedData containing the peaks
outfile	A character vector containing the path to the file to write the summary output.
append	TRUE or FALSE, default FALSE

Value

A data frame with 3 columns as n.peaksWithPattern (number of peaks with the pattern), n.totalPeaks (total number of peaks in the input) and Pattern (the corresponding pattern).

Author(s)

Lihua Julie Zhu

Examples

```
peaks = RangedData(IRanges(start=c(100, 500), end=c(300, 600),
                          names=c("peak1", "peak2")),
                  space=c("NC_008253", "NC_010468"))
filepath =system.file("extdata", "examplePattern.fa", package="ChIPpeakAnno")
library(BSgenome.Ecoli.NCBI.20080805)
summarizePatternInPeaks(patternFilePath=filepath, format="fasta",
                       skip=0L, BSgenomeName=Ecoli, peaks=peaks)
```

toGRanges

Convert dataset to GRanges

Description

Convert BED, GFF, RangedData or any user defined dataset to GRanges

Usage

```
toGRanges(data, format=c("BED", "GFF", "RangedData", "MACS", "others"),
          header=FALSE, comment.char="#", colNames=NULL, ...)
```

Arguments

data	BED, GFF, RangedData or any user defined dataset or their file path.
format	data format. If the data format is set to BED or GFF, please refer to http://genome.ucsc.edu/FAQ/FAQformat for column order. or MACS output file.
header	a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns.
comment.char	character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.
colNames	If the data format is set to "others", colname must be defined. And the colname must contain space, start and end. If your column is names as seqname or chrom, and so on, please rename it as space.
...	parameters passed to read.table

Value

An object of [GRanges](#)

Author(s)

Jianhong Ou

Examples

```
rd <- RangedData(IRanges(start = c(967654, 2010897, 2496704),
  end = c(967754, 2010997, 2496804), names = c("Site1", "Site2", "Site3")),
  space = c("1", "2", "3"), strand=as.integer(1),feature=c("a","b","f"))
toGRanges(rd, format="RangedData")
```

translatePattern	<i>translate pattern from IUPAC Extended Genetic Alphabet to regular expression</i>
------------------	---

Description

translate pattern containing the IUPAC nucleotide ambiguity codes to regular expression. For example, Y->[C|T], R-> [A|G], S-> [G|C], W-> [A|T], K-> [T|U|G], M-> [A|C], B-> [C|G|T], D-> [A|G|T], H-> [A|C|T], V-> [A|C|G] and N-> [A|C|T|G].

Usage

```
translatePattern(pattern)
```

Arguments

pattern a character vector with the IUPAC nucleotide ambiguity codes

Value

a character vector with the pattern represented as regular expression

Author(s)

Lihua Julie Zhu

See Also

countPatternInSeqs, summarizePatternInPeaks

Examples

```
pattern1 = "AACCNWМК"
translatePattern(pattern1)
```

TSS.human.GRCh37 *TSS annotation for human sapiens (GRCh37) obtained from biomaRt*

Description

TSS annotation for human sapiens (GRCh37) obtained from biomaRt

Usage

```
data(TSS.human.GRCh37)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

```
description description of the gene
```

Details

used in the examples Annotation data obtained by:

```
mart = useMart(biomart = "ENSEMBL_MART_ENSEMBL", host="grch37.ensembl.org", path="/biomart/martservice",
dataset = "hsapiens_gene_ensembl")
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.human.GRCh37)
slotNames(TSS.human.GRCh37)
```

TSS.human.GRCh38 *TSS annotation for human sapiens (GRCh38) obtained from biomaRt*

Description

TSS annotation for human sapiens (GRCh38) obtained from biomaRt

Usage

```
data(TSS.human.GRCh38)
```

Format

Formal class 'GRanges' [package "GenomicRanges"] with ensembl id as names.

Details

used in the examples Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")
getAnnotation(mart, featureType = "TSS", output="GRanges")
```

Examples

```
data(TSS.human.GRCh38)
slotNames(TSS.human.GRCh38)
```

TSS.human.NCBI36

TSS annotation for human sapiens (NCBI36) obtained from biomaRt

Description

TSS annotation for human sapiens (NCBI36) obtained from biomaRt

Usage

```
data(TSS.human.NCBI36)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

```
description description of the gene
```

Details

used in the examples Annotation data obtained by:

```
mart = useMart(biomart = "ensembl_mart_47", dataset = "hsapiens_gene_ensembl", archive=TRUE)
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.human.NCBI36)
slotNames(TSS.human.NCBI36)
```

TSS.mouse.GRCm38	<i>TSS annotation data for Mus musculus (GRCm38.p1) obtained from biomaRt</i>
------------------	---

Description

TSS annotation data for Mus musculus (GRCm38.p1) obtained from biomaRt

Usage

```
data(TSS.mouse.GRCm38)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

```
description description of the gene
```

Details

Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "mmusculus_gene_ensembl")
```

```
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.mouse.GRCm38)
slotNames(TSS.mouse.GRCm38)
```

TSS.mouse.NCBIM37	<i>TSS annotation data for mouse (NCBIM37) obtained from biomaRt</i>
-------------------	--

Description

TSS annotation data for mouse (NCBIM37) obtained from biomaRt

Usage

```
data(TSS.mouse.NCBIM37)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

description description of the gene

Details

Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "mmusculus_gene_ensembl")
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.mouse.NCBIM37)
slotNames(TSS.mouse.NCBIM37)
```

TSS.rat.RGSC3.4

TSS annotation data for rat (RGSC3.4) obtained from biomaRt

Description

TSS annotation data for rat (RGSC3.4) obtained from biomaRt

Usage

```
data(TSS.rat.RGSC3.4)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

description description of the gene

Details

Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "rnorvegicus_gene_ensembl")
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.rat.RGSC3.4)
slotNames(TSS.rat.RGSC3.4)
```

TSS.rat.Rnor_5.0	<i>TSS annotation data for Rattus norvegicus (Rnor_5.0) obtained from biomaRt</i>
------------------	---

Description

TSS annotation data for Rattus norvegicus (Rnor_5.0) obtained from biomaRt

Usage

```
data(TSS.rat.Rnor_5.0)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

```
description description of the gene
```

Details

Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "rnorvegicus_gene_ensembl")
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.rat.Rnor_5.0)
slotNames(TSS.rat.Rnor_5.0)
```

TSS.zebrafish.Zv8	<i>TSS annotation data for zebrafish (Zv8) obtained from biomaRt</i>
-------------------	--

Description

TSS annotation data for zebrafish (Zv8) obtained from biomaRt

Usage

```
data(TSS.zebrafish.Zv8)
```


Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

description description of the gene

Details

Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "drerio_gene_ensembl")
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.zebrafish.Zv8)
slotNames(TSS.zebrafish.Zv8)
```

TSS.zebrafish.Zv9	<i>TSS annotation for Danio rerio (Zv9) obtained from biomaRt</i>
-------------------	---

Description

TSS annotation for Danio rerio (Zv9) obtained from biomaRt

Usage

```
data(TSS.zebrafish.Zv9)
```

Format

GRanges with slot start holding the start position of the gene, slot end holding the end position of the gene, slot names holding ensembl gene id, slot seqnames holding the chromosome location where the gene is located and slot strand holding the strand information. In addition, the following variables are included.

description description of the gene

Details

Annotation data obtained by:

```
mart = useMart(biomart = "ensembl", dataset = "drerio_gene_ensembl")
getAnnotation(mart, featureType = "TSS")
```

Examples

```
data(TSS.zebrafish.Zv9)
slotNames(TSS.zebrafish.Zv9)
```

`write2FASTA`*write sequences to a file in fasta format*

Description

write the sequences obtained from `getAllPeakSequence` to a file in fasta format leveraging `writeFASTA` in `Biostrings` package. FASTA is a simple file format for biological sequence data. A FASTA format file contains one or more sequences and there is a header line which begins with a `>` preceding each sequence.

Usage

```
write2FASTA(mySeq, file="", width=80)
```

Arguments

<code>mySeq</code>	RangedData with variables name and sequence ,e.g., results obtained from <code>getAllPeakSequence</code>
<code>file</code>	Either a character string naming a file or a connection open for reading or writing. If "" (the default for <code>write2FASTA</code>), then the function writes to the standard output connection (the console) unless redirected by <code>sink</code>
<code>width</code>	The maximum number of letters per line of sequence

Value

Output as FASTA file format to the naming file or the console.

Author(s)

Lihua Julie Zhu

Examples

```
peaksWithSequences = RangedData(IRanges(start=c(1000, 2000), end=c(1010, 2010)),
names=c("id1", "id2"), sequence= c("CCCCCCCCGGGG", "TTTTTTTAAAAA"))
write2FASTA(peaksWithSequences, file="testseq.fasta", width=50)
```

Index

- *Topic **\textasciitildekwd1**
 - [assignChromosomeRegion](#), [11](#)
- *Topic **\textasciitildekwd2**
 - [assignChromosomeRegion](#), [11](#)
- *Topic **datasets**
 - [annotatedPeak](#), [7](#)
 - [enrichedGO](#), [19](#)
 - [ExonPlusUtr.human.GRCh37](#), [21](#)
 - [myPeakList](#), [36](#)
 - [Peaks.Ste12.Replicate1](#), [37](#)
 - [Peaks.Ste12.Replicate2](#), [37](#)
 - [Peaks.Ste12.Replicate3](#), [38](#)
 - [TSS.human.GRCh37](#), [44](#)
 - [TSS.human.GRCh38](#), [44](#)
 - [TSS.human.NCBI36](#), [45](#)
 - [TSS.mouse.GRCm38](#), [46](#)
 - [TSS.mouse.NCBIM37](#), [46](#)
 - [TSS.rat.RGSC3.4](#), [47](#)
 - [TSS.rat.Rnor_5.0](#), [48](#)
 - [TSS.zebrafish.Zv8](#), [48](#)
 - [TSS.zebrafish.Zv9](#), [49](#)
- *Topic **graph**
 - [makeVennDiagram](#), [34](#)
- *Topic **misc**
 - [addAncestors](#), [4](#)
 - [addGeneIDs](#), [5](#)
 - [annotatePeakInBatch](#), [8](#)
 - [BED2RangedData](#), [13](#)
 - [binOverFeature](#), [14](#)
 - [condenseMatrixByColnames](#), [16](#)
 - [convert2EntrezID](#), [17](#)
 - [countPatternInSeqs](#), [18](#)
 - [egOrgMap](#), [19](#)
 - [findOverlappingPeaks](#), [22](#)
 - [findOverlapsOfPeaks](#), [24](#)
 - [findVennCounts](#), [25](#)
 - [getAllPeakSequence](#), [26](#)
 - [getAnnotation](#), [27](#)
 - [getEnrichedGO](#), [28](#)
 - [getEnrichedPATH](#), [30](#)
 - [getVennCounts](#), [32](#)
 - [GFF2RangedData](#), [33](#)
 - [peaksNearBDP](#), [39](#)
 - [summarizePatternInPeaks](#), [41](#)
 - [toGRanges](#), [42](#)
 - [translatePattern](#), [43](#)
 - [write2FASTA](#), [50](#)
- *Topic **package**
 - [ChIPpeakAnno-package](#), [3](#)
 - [addAncestors](#), [4](#)
 - [addGeneIDs](#), [5](#), [10](#)
 - [annotatedPeak](#), [7](#)
 - [annotatePeakInBatch](#), [8](#), [25](#)
 - [AnnotationDbi](#), [6](#)
 - [assignChromosomeRegion](#), [11](#)
 - [BED2RangedData](#), [13](#)
 - [binOverFeature](#), [14](#)
 - [ChIPpeakAnno \(ChIPpeakAnno-package\)](#), [3](#)
 - [ChIPpeakAnno-deprecated](#), [15](#)
 - [ChIPpeakAnno-package](#), [3](#)
 - [condenseMatrixByColnames](#), [16](#)
 - [convert2EntrezID](#), [17](#)
 - [countPatternInSeqs](#), [18](#)
 - [Deprecated](#), [16](#)
 - [egOrgMap](#), [19](#)
 - [enrichedGO](#), [19](#)
 - [ExonPlusUtr.human.GRCh37](#), [21](#)
 - [findOverlappingPeaks](#), [10](#), [22](#), [25](#), [33](#), [35](#)
 - [findOverlappingPeaks-deprecated \(findOverlappingPeaks\)](#), [22](#)
 - [findOverlapsOfPeaks](#), [16](#), [22](#), [24](#)
 - [findVennCounts](#), [25](#)
 - [getAllPeakSequence](#), [26](#)

getAnnotation, 27
getBM, 6
getEnrichedGO, 28
getEnrichedPATH, 30
getVennCounts, 25, 32
GFF2RangedData, 33
GRanges, 8, 9, 16, 22, 24, 26–28, 32, 34, 39,
41, 42

listAttributes(mart), 6
listFilters(mart), 6

makeVennDiagram, 10, 25, 33, 34
myPeakList, 36

Peaks.Ste12.Replicate1, 37
Peaks.Ste12.Replicate2, 37
Peaks.Ste12.Replicate3, 38
peaksNearBDP, 10, 39

RangedData, 8, 9, 16, 22, 24, 26–28, 32, 34,
39, 41
read.table, 42

summarizePatternInPeaks, 10, 41

toGRanges, 42
translatePattern, 43
TSS.human.GRCh37, 44
TSS.human.GRCh38, 44
TSS.human.NCBI36, 45
TSS.mouse.GRCm38, 46
TSS.mouse.NCBIM37, 46
TSS.rat.RGSC3.4, 47
TSS.rat.Rnor_5.0, 48
TSS.zebrafish.Zv8, 48
TSS.zebrafish.Zv9, 49
TxDb, 11, 12

useMart, 5

venn.diagram, 35

write2FASTA, 50