ChIPpeakAnno

April 20, 2011

`addAncestors` *Add GO ids of the ancestors for a given vector of GO ids*

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

annotatedPeak *Annotated Peaks*

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

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.

`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

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

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, y
}
```

`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"), An
```

Arguments

`myPeakList`	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 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(GO.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`	nearestStart: will output the nearest features calculated as peak start - feature start (feature end if feature resides at minus strand); overlapping: will output overlapping features with maximum gap =0 between peak range and feature range; both: will output all the nearest features, in addition, will output any features that overlap the peak that is not the nearest features.
`multiple`	not applicable when output is nearestStart. TRUE: output multiple overlapping features for each peak. FALSE: output at most one overlapping feature for each peak
`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. To be compatible with previous version, by default using TSS

Value

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 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

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

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)
as.data.frame(annotatedPeak)
```

```
## example 2: you have a list of transcription factor biding sites from literature and ar
## 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 Rang
## of the binding site, end is the end of the binding site, names is the name of the binc
## space and strand are the chromosome name and strand where the binding site is located.

myexp =  RangedData(IRanges(start=c(1543200,1557200,1563000,1569800,167889600,100,1000),e
literature = RangedData(IRanges(start=c(1549800,1554400,1565000,1569400,167888600,120,800
annotatedPeak1= annotatePeakInBatch(myexp, AnnotationData = literature)
pie(table(as.data.frame(annotatedPeak1)$insideFeature))
as.data.frame(annotatedPeak1)
### use BED2RangedData or GFF2RangedData to convert BED format or GFF format to RangedDat
test.bed = data.frame(cbind(chrom = c("4", "6"), chromStart=c("100", "1000"),chromEnd=c("
test.rangedData = BED2RangedData(test.bed)
annotatePeakInBatch(test.rangedData, AnnotationData = literature)
test.GFF = data.frame(cbind(seqname  = c("chr4", "chr4"), source=rep("Macs", 2), feature=
test.rangedData = GFF2RangedData(test.GFF)
as.data.frame(annotatePeakInBatch(test.rangedData, AnnotationData = literature))
}
```

`BED2RangedData` *convert BED format to RangedData*

Description

convert BED format to RangedData

Usage

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

Arguments

`data.BED`	BED format data frame, please refer to http://genome.ucsc.edu/FAQ/FAQformat#format1 for details
`header`	TRUE or FALSE, default to FALSE, indicates whether data.BED file has BED header

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 formated 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("
test.rangedData = BED2RangedData(test.bed)
```

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:	1.3.6
Date:	2009-03-20
License:	LGPL
LazyLoad:	yes

Author(s)

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

Maintainer: 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)

Examples

```
if (interactive())
{
data(myPeakList)
        data(TSS.human.NCBI36)

myPeakList1 = myPeakList[1:6,]

annotatedPeak = annotatePeakInBatch(myPeakList1, AnnotationData=TSS.human.NCBI36)

peaks = RangedData(IRanges(start=c(100, 500), end=c(300, 600), names=c("peak1", "peak2"))
library(BSgenome.Ecoli.NCBI.20080805)

peaksWithSequences = getAllPeakSequence(peaks, upstream = 20,
downstream = 20, genome = Ecoli)

write2FASTA(peaksWithSequences, file="testseq.fasta", width=50)

library(org.Hs.eg.db)
enrichedGO = getEnrichedGO(annotatedPeak, orgAnn ="org.Hs.eg.db", maxP=0.01, multiAdj=FAI

enriched.biologicalprocess = enrichedGO$bp
enriched.molecularfunction = enrichedGO$mf
enriched.cellularcomponent = enrichedGO$cc

data(annotatedPeak)
y = annotatedPeak$distancetoFeature[!is.na(annotatedPeak$distancetoFeature)]
hist(y, xlab="Distance To Nearest TSS", main="", breaks=1000, xlim=c(min(y)-100, max(y)+1
}
```

`convert2EntrezID` *Convert other common IDs such as ensemble gene id, gene symbol, refseq id to entrez gene ID.*

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 ensemble_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
```

---

enrichedGO *Enriched Gene Ontology terms used as example*

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 = 100, multiple = c(TRUE, FALSE), Na
```

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.
`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.
`NameOfPeaks1`	Name of the Peaks1, used for generating column name.
`NameOfPeaks2`	Name of the Peaks2, used for generating column name.

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(1555
peaks2 =  RangedData(IRanges(start=c(1549800,1554400,1565000,1569400,167888600),end=c(155
t1 =findOverlappingPeaks(peaks1, peaks2, maxgap=1000, multiple=F,NameOfPeaks1="TF1", Name
r = t1$OverlappingPeaks
pie(table(r$overlapFeature))
as.data.frame(t1$MergedPeaks)
}
```

---

findVennCounts *Obtain Counts for Venn Diagram, internal function for makeVennDigram*

Description

Obtain Counts for two peak ranges, internal function for makeVennDigram

Usage

```
findVennCounts(Peaks, NameOfPeaks, maxgap = 0, totalTest)
```

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.
totalTest	Numeric value to specify the total number of tests performed to obtain the list of peaks.

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)) 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)) findVennCounts(RangedDataList(peaks1,peaks2), NameOfPeaks=c("TF1", "TF2"), maxgap=0,totalTest=100)

Author(s)

Lihua Julie Zhu

See Also

makeVennDiagram

---

getAllPeakSequence *Obtain genomic sequences around the peaks*

Description

Obtain genomic sequences around the peaks leveraging BSgenome and biomaRt package

Usage

```
getAllPeakSequence(myPeakList, upstream = 200, downstream = 200, genome, Annotat
```

Arguments

myPeakList	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 getAnnotation with featureType="TSS". For example, data(TSS.human.NCBI36),data(TSS.mouse.NCBIM data(GO.rat.RGSC3.4) and data(TSS.zebrafish.Zv8). If not supplied, then annotation will be obtained from biomaRt automatically using the mart object

Value

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

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 = RangedData(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)
```

---

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", "ExonPl
```

Arguments

mart	mart object, see useMart of bioMaRt package for details
featureType	TSS, miRNA, Exon, 5'UTR, 3'UTR or Exon plus UTR

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
description	description of the feeature 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

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=0.0
```

Arguments

annotatedPeak	RangedData 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, TURE 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	enriched biological process with the following 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

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

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, 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

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=
test.rangedData = GFF2RangedData(test.GFF)
```

---

makeVennDiagram *Make Venn Diagram from two peak ranges*

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=0, totalTest, cex = 1.5, counts.col =
```

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.
`totalTest`	Numeric value to specify the total number of tests performed to obtain the list of peaks.
`cex`	Numerical value giving the amount by which the contrast names should be scaled on the plot relative to the default.plotting text. See par.
`counts.col`	optional vector of color specifications defining the colors by which the circles should be drawn. See par.

Details

This is a wrapper function for vennDiagram from limma package.

Value

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

Author(s)

Lihua Julie Zhu

See Also

findOverlappingPeaks

Examples

```
if (interactive())
{
peaks1 = RangedData(IRanges(start = c(967654, 2010897, 2496704), end = c(967754, 2010997,
peaks2 = RangedData(IRanges(start = c(967659, 2010898,  2496700, 3075866, 3123260), end =
makeVennDiagram(RangedDataList(peaks1,peaks2), NameOfPeaks=c("TF1", "TF2"), totalTest=100
}
```

---

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 Lefranois, 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 Lefranois, 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 Lefranois, 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)
```

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

RangedData with slot start holding the start position of the gene, slot end holding the end position of the gene, slot rownames holding ensembl gene 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 gene

Details

used in the examples Annotation data obtained by:

mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")

getAnnotation(mart, featureType = "TSS")

Examples

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

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

RangedData with slot start holding the start position of the gene, slot end holding the end position of the gene, slot rownames holding ensembl gene 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 gene

Details

used in the examples Annotation data obtained by:

mart = useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")

getAnnotation(mart, featureType = "TSS")

Examples

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

TSS.mouse.NCBIM37 *TSS annotation data for mouse (NCBIM37) obtained from biomaRt*

Description

TSS annotation data for mouse (NCBIM37) obtained from biomaRt

Usage

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

Format

RangedData with slot start holding the start position of the gene, slot end holding the end position of the gene, slot rownames holding ensembl gene 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 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

RangedData with slot start holding the start position of the gene, slot end holding the end position of the gene, slot rownames holding ensembl gene 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 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.zebrafish.Zv8 *TSS annotation data for zebrafish (Zv8) obtained from biomaRt*

Description

TSS annotation data for zebrafish (Zv8) obtained from biomaRt

Usage

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

Format

RangedData with slot start holding the start position of the gene, slot end holding the end position of the gene, slot rownames holding ensembl gene 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 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)
```

---

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 > proceeding each sequence.

Usage

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

Arguments

mySeq	RangedData with varibles name and sequence ,e.g., results obtained from getAllPeakSequence
file	Either a character string naming a file or a connection open for reading or writing. If "" (the default for write2FASTA), then the function writes to the standard output connection (the console) unless redirected by sink
width	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("
write2FASTA(peaksWithSequences, file="testseq.fasta", width=50)
```

Index