Package 'rtfbsdb'

March 21, 2016

Version 0.3.5

Date 2016-03-08

Title An Integrated Framework for Transcription Factor Binding Site Analysis

Author Charles G. Danko <dankoc@gmail.com>, Zhong Wang<zw355@cornell.edu>, Andre L. Martins<alm253@cornell.edu>

Maintainer Zhong Wang<zw355@cornell.edu>

Depends R (>= 2.6)

Imports rphast, rtfbs, parallel, grid, cluster, methods, latticeExtra, lattice, apcluster, vioplot, tools

LinkingTo

Suggests RCurl, stringr, bigWig, MotifDb

Description This package finds the transcription factor binding sites within specified genome loci and estimate the enrichment between two case-control group of different genomic loci based on a scoring algorithm driven by a Hidden Markov model and the CisBP database (or other data source, such as Jaspar, Transfac). It clusters motifs with similar DNA sequence specificities and optionally integrates RNA-seq or PRO-seq data to restrict analyses to motifs recognized by TFs expressed in the cell type of interest. PDF report is provided for the motif clustering, tfbs scaning and enrichment test.

License GPL-3

biocViews Sequencing, Analysis

LazyLoad yes

R topics documented:

`CisBP.db-class` *Class* `"CisBP.db"`

Description

The motif library from CisBP web site.
Link: `http://cisbp.ccbr.utoronto.ca/`

Objects from the Class

Objects can be created by calls of the form `CisBP.extdata`, `CisBP.zipload`, `CisBP.download`.

Slots

`species`: String indicating the species name defined in the CisBP dataset.

`zip.file`: String indicating the filename of temporary data file.

`zip.url`: String indicating the download source.

`zip.date`: String indicating the download date.

`file.tfinfo`: String indicating the TF filename, defulat is TF_Information.txt.

Extends

Class `"tfbs.db"`, directly.

Methods

tfbs.createFromCisBP Build a tfbs object by querying the meta file of CisBP dataset and subsetting the results.

CisBP.group Get the statistical summary by grouping the field in the CisBP dataset.

CisBP.getTFinformation Get the TF Information stored in the CisBP dataset.

References

Weirauch, M. T., Yang, A., Albu, M., Cote, A. G., Montenegro-Montero, A., Drewe, P., ... & Hughes, T. R. (2014). Determination and inference of eukaryotic transcription factor sequence specificity. Cell, 158(6), 1431-1443.

See Also

`CisBP.getTFinformation`, `CisBP.group`, `tfbs.createFromCisBP`

Examples

```
showClass("CisBP.db")
```

`CisBP.download` *Download CisBP dataset.*

Description

Download TF data file from CisBP dataset and store it to temporary folder

Usage

```
CisBP.download(species = "Homo_sapiens",
      url = "http://cisbp.ccbr.utoronto.ca/bulk_archive.php")
```

Arguments

`species`	String, indicating the species name in the CisBP dataset
`url`	String, the URL of bulk dowbnloads from CisBP dataset, default is `http://cisbp.ccbr.utoronto.ca/bulk_archive.php`

Details

The dowload function has been confirmed in the web site of *cisbp.ccbr.utoronto.ca* o June, 2015.

Value

A CisBP object (class name: `"CisBP.db"`) is returned with four items:

`species`	String indicating the species name
`zip.file`	String indicating the filename of temporary data file.
`zip.url`	String indicating the download source
`file.tfinfo`	String indicating the TF filename, default is TF_Information.txt.

References

Weirauch, M. T., Yang, A., Albu, M., Cote, A. G., Montenegro-Montero, A., Drewe, P., ... & Hughes, T. R. (2014). Determination and inference of eukaryotic transcription factor sequence specificity. Cell, 158(6), 1431-1443.

See Also

See Also as CisBP.ziploadCisBP.extdata.

Examples

```
#download zebra fish dataset
db1 <- CisBP.download("Danio_rerio");

#download Felis_catus dataset
db2 <- CisBP.download("Felis_catus");
```

CisBP.extdata *Load internal CisBP dataset.*

Description

Build a CisBP object from the internal zip file stored in this package

Usage

```
CisBP.extdata(species)
```

Arguments

species	String, only valid for human and mouse species, i.e. Homo_sapiens, Mus_musculus, or Drosophila_melanogaster

Details

The CisBP data for Homo_sapiens and Mus_musculus are delivered by this package. When you use the newest dataset, you should download it from the website by CisBP.download.

Value

A CisBP object (class name: "CisBP.db") is returned with four items:

species	String indicating the species name defined in the CisBP dataset.
zip.file	String indicating the filename of temporary data file.
zip.url	String indicating the download source
file.tfinfo	String indicating the TF filename, default is TF_Information.txt.

See Also

See Also as CisBP.ziploadCisBP.download.

Examples

```
#reading human data from extension data file in the package
db.human <- CisBP.extdata("Homo_sapiens")

#reading Drosophila_melanogaster from extension data file in the package
db.dm3 <- CisBP.extdata("dm3")
```

`CisBP.getTFinformation` *Get TF information with PWM status*

Description

Get TF information with PWM status

Usage

```
CisBP.getTFinformation(cisbp.db, tf.information.type = NA)
```

Arguments

`cisbp.db`	A CisBP object (`"CisBP.db"`) including the TF_Information.txt.
`tf.information.type`	Number, indicating which TF meta file will be used. Available values are 1 for TF_Information.txt, 2 for TF_Information_all_motifs.txt and 3 for F_Information_all_motifs_plus.txt.

Details

Three TF information files in CisBP dataset.

1: TF_Information.txt : (direct motifs) or (no direct but inferred motifs with 90%)
2: TF_Information_all_motifs.txt: (direct motifs) and (inferred motifs above the threshold)
3: F_Information_all_motifs_plus.txt: All motifs

The following parts are copied from RAEDME.txt in zipped CisBP data file.

TF_Information.txt, TF_Information_all_motifs.txt, TF_Information_all_motifs_plus.txt - These files contain information on the TFs.

'TF_Information.txt' contains, for each TF, all directly determined motifs (see below). If a TF does not have a directly determined motif, this file will also include its best inferred motif. 'Best' is defined as the motif(s) obtained from the most similar TF (based on the

'TF_Information_all_motifs.txt' is a superset of 'TF_Information.txt'. It also includes any motif that can be inferred for a given TF, given the TF family-specific threshold. For example, if a TF has a directly determined motif, and two TFs with motifs with 90 TF_Information_all_motifs.txt will include all three motifs. Likewise, if a TF does not have a direct motif, but has two TFs with 90

'TF_Information_all_motifs_plus.txt' is a superset of the other two files. It contains all motifs for a given TF, which includes all direct motifs, and all inferred motifs above the threshold.

Value

A data frame returned with the status indicating PWM data is existing or not

TF_ID	Internal CisBP ID for the TF. Each gene has a unique TF_ID
Family_ID	Internal CisBP ID for the TF family. A family is the unique set of DNA binding domains (DBDs) present in the protein.
TSource_ID	Internal CisBP ID for the source of the TF (i.e. where its genome sequence was obtained).
Motif_ID	Internal CisBP ID for the associated motif.
MSource_ID	Internal CisBP ID for the source of the motif (i.e. which database or study it came from)
DBID	External ID of the RBP (e.g., Ensembl ID)
TF_Name	Name of the TF
TF_Species	Species of the TF
TF_Status	Motif status of the TF. 'D' stands for directly determined motif. 'I' indicates that the motif is inferred from another TF, based on DBD similarity (see Weirauch et al. 2013 for details). 'N' means no motif is available.
Family_Name	Name of the TF's family
DBDs	The unique set of DBDs (Pfam names) present in the TF
DBD_Count	Number of unique DBDs in the TF
Cutoff	Cutoff used to infer motifs for the TF family
DBID	Motif ID from the associated database or study
Motif_Type	Experimental assay used to determine the motif
MSource_Identifier	ID for the source of the motif (i.e., its project name)
MSource_Type	Internal CisBP ID for the motif category
MSource_Author	First author for the source of the motif
MSource_Year	Year of publication of the motif source
PMID	Pubmed ID of the motif source
MSource_Version	Version of the source (i.e. database build)
TFSource_Name	Source of the TF (i.e. where did the genome build come from?)
TFSource_URL	URL of the TF source
TFSource_Year	Year the genome data was downloaded
TFSource_Month	Month the genome data was downloaded
TFSource_Day	Day the genome data was downloaded
motif_existing	Status indicating PWM data is existing or not

See Also

See Also as CisBP.group, CisBP.extdata, CisBP.ziploa, CisBP.download

Examples

```
# Load the internal CisBP dataset
db_human <- CisBP.extdata("Homo_sapiens");

df.tfinfo <- CisBP.getTFinformation( db_human, tf.information.type = 2)
show(head(df.tfinfo));
```

`CisBP.group` *Summarize the motif number.*

Description

Get the statistical summary by grouping the field in the CisBP dataset.

Usage

```
CisBP.group(cisbp.db,
      group.by=c("tf_name", "tf_species", "tf_status", "family_name",
            "motif_type", "msource_id"),
      tf.information.type=NA )
```

Arguments

`cisbp.db`	A CisBP object (`"CisBP.db"`) including the TF_Information.txt.
`group.by`	String, indicating which field will be used to group values. Available values are tf_name, tf_species, tf_status, family_name, motif_type and msource_id.
`tf.information.type`	Number, indicating which TF meta file will be used. Available values are 1 for TF_Information.txt, 2 for TF_Information_all_motifs.txt and 3 for F_Information_all_motifs_plus.txt.

Details

Three TF information files in CisBP dataset.

1: TF_Information.txt : (direct motifs) or (no direct but inferred motifs with 90%)
2: TF_Information_all_motifs.txt: (direct motifs) and (inferred motifs above the threshold)
3: F_Information_all_motifs_plus.txt: All motifs

Value

A data frame returned includes two columns

`group_by`	Values of grouping field
`number`	Counts of group value

See Also

See Also as `tfbs.createFromCisBP`

Examples

```
# Load the internal CisBP dataset
db_human <- CisBP.extdata("Homo_sapiens");

# Group the motif count by the column of family_name in TF_Information.txt
gr1 <- CisBP.group(db_human, group.by="family_name", tf.information.type=1 );

# Group the motif count by the column of tf_status in TF_Information.txt
gr2 <- CisBP.group(db_human, group.by="tf_status", tf.information.type=1 );

# Group the motif count by the column of tf_status
# in TF_Information_all_motifs.txt
gr3 <- CisBP.group(db_human, group.by="tf_status", tf.information.type=2);

# Group the motif count by the column of tf_status
# in F_Information_all_motifs_plus.txt
gr4 <- CisBP.group(db_human, group.by="tf_status", tf.information.type=3);
```

`CisBP.zipload` *Load the zipped CisBP file.*

Description

Build a CisBP object from the zipped CisBP file.

Usage

```
CisBP.zipload(zip.file, species = "Homo_sapiens")
```

Arguments

`zip.file`	String, indicating the zipped file data
`species`	String, indicating the species name in the CisBP database

Details

The zip data canbe downloaded from the web site, please check `CisBP.download`.

Value

A CisBP object (class name: `"CisBP.db"`) is returned with four items:

`species`	String indicating the species name
`zip.file`	String indicating the filename of temporary data file.
`zip.url`	String indicating the download source
`file.tfinfo`	String indicating the TF filename, default is TF_Information.txt.

See Also

See Also as `CisBP.extdata`, `CisBP.download`.

Examples

```
# Download the dataset
db1 <- CisBP.download("Arabidopsis_thaliana");

# Loading the zip file, the db2 and db3 have same TF data.
# Here is an example to show how to use CisBP.ziploaD.
# We dont nee to download it by CisBP.download and then load it
# by CisBP.ziploaD
db2 <- CisBP.ziploaD(db1@zip.file, species="Arabidopsis thaliana");
```

---

print.tfbs.enrichment *Print the brief enrichment results*

Description

Print the brief enrichment results.

Usage

```
## S3 method for class 'tfbs.enrichment'
print(x, ..., pv.threshold=0.05, pv.adj=NA )
```

Arguments

x	The result obtained by tfbs.enrichmentTest.
...	Additional arguments affecting the print produced.
pv.threshold	Numeric value,indicating whether the different cutoff of p-value is applied to select the significant motifs.
pv.adj	String, P-values correct method for p.adjust function. The available values are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr" or "none". (default="bonferroni").

Details

This command shows the calling parameters and significant motifs from the result object.
The significant motifs are selected by the corrected p-value cutoff(0.05) and at most 20 significant motifs are listed. The adjust method of p-value is defined in the calling function.

Value

No return values.

See Also

See also as tfbs.enrichmentTest.

Examples

```
#See example in tfbs.enrichmentTest
```

print.tfbs.finding *Print scanning result of TF sites.*

Description

Print scanning result of TF sites.

Usage

```
## S3 method for class 'tfbs.finding'
print(x, ...)
```

Arguments

x	The result obtained by tfbs.scanTFsite.
...	Additional arguments affecting the print produced.

Details

This function shows a brief information including calling parameters and enriched motifs.

Value

No return values.

See Also

See Also as tfbs.scanTFsite

Examples

```
#See example in tfbs.scanTFsite
```

summary.tfbs.enrichment *Summarize the enrichment result*

Description

Return the significant motifs based on the adjust p-values using multiple comparisons.

Usage

```
## S3 method for class 'tfbs.enrichment'
summary(object, pv.threshold = 0.05, pv.adj = NA, ...)
```

Arguments

object	The result obtained by tfbs.enrichmentTest.
pv.threshold	The p-value threshold for significant motifs.
pv.adj	P-values adjust method for p.adjust function. The available values are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr" or "none".
...	Additional arguments affecting the summary produced.

Details

A data frame with 6 colums is returned.

Value

The results is a data frame including 6 columns,

motif.id	Motif ID
tf.name	TF Name
Npos	Read count in positive loci.
expected	Read count in negative loci.
fe.ratio	The ratio of read counts between positive loci and negative loci.
starch	Cpmporessed Bed filename
pvalue	p-value
pv.adj	adjusted p-value by multiple comparson method.

See Also

See also as tfbs.enrichmentTest.

---

summary.tfbs.finding *Summarize scanning results.*

Description

Return a data frame with summarized TF sites for every motif if the calling parameter is "matches".

Usage

```
## S3 method for class 'tfbs.finding'
summary(object, ...)
```

Arguments

object	The result obtained by tfbs.scanTFsite.
...	Additional arguments affecting the summary produced.

Details

summary in class of tfbs.finding is returned.

Value

This function will return a data frame with summarized TF sites for every motif if the calling parameter is "matches", otherwise, NULL will be returned.

See Also

See Also as `tfbs.scanTFsite`

`tfbs` *Create a tfbs object from the supplied PWM files.*

Description

Create a tfbs object from the supplied PWM files.

Usage

```
tfbs(filenames,
      names,
      species="Homo_sapiens",
      tf_info = NULL,
      tf_missing = NULL, ...)
```

Arguments

`filenames`	Vector of PWM files
`names`	Vector of unique gene symbols.
`species`	String indicating species name
`tf_info`	Data frame including meta information copied from CisBP data file for all existing motifs., Default: NULL
`tf_missing`	Data frame including meta information copied from CisBP data file for missing motifs., Default: NULL
`...`	Parameters,such as pseudocount, force_even, and the parameters used in read.table function.

Details

Load the PWM files to build a `"tfbs"` object.

Value

A tfbs object (class: `"tfbs"`) including all PWM matrics.The all attributes are as follows:

`TFID`	Vector of non-unique ID for TF.
`species`	String indicating the species name
`ntfs`	Number of motifs in matrix.
`pwm`	A list including PWM matics.
`filename`	Vector of PWM filename.

mgisymbols	Unique gene symbols for TF.
tf_info	Data frame, including extra information for all existing PWMs, it maybe different with motif dataset, default:NULL.
tf_missing	Data frame, including extra information for missing PWMs, it maybe different with motif dataset, default:NULL.
distancematrix	Distance matrix between motifs returned by tfbs.clusterMotifs, default:NULL.
expressionlevel	Data frame indicatig the result of expression level returned by tfbs.getExpression, default:NULL.
cluster	Matrix with 2 columns returned by tfbs.clusterMotifs, 1st column is the index of motifs and 2nd column is the group number of clustering, default:NULL.

The tfbs object can be created by the function of tfbs, tfbs.dirs, tfbs.createFromCisBP.

See Also

tfbs, tfbs.dirs, tfbs.createFromCisBP

Examples

```
# M3590_1.01 PAX5 ENSG00000196092
# M3590_1.01 PAX5 ENSG00000196092
fs1 <- system.file("extdata","M3590_1.01.pwm", package="rtfbsdb")
fs2 <- system.file("extdata","M3591_1.01.pwm", package="rtfbsdb")

cat(fs1, "\n");

tfs <- tfbs( c( fs1, fs2 ), names=c("M3590_1.01","M3591_1.01"),
      header=TRUE, sep="\t" , row.names=1 );
str(tfs);
```

---

tfbs-class *Class* "tfbs"

Description

Tfbs object is a collection of motif PWM data. Some functions are provided based on the PWM and GENCODE data, such as clustering, search and compare.

Objects from the Class

Objects can be created by calls of the function of tfbs.createFromCisBP, tfbs.dirs and tfbs.

Slots

species String indicating the species name

ntfs Number of motifs in matrix.

pwm A list including PWM matics.

filename Vector of PWM filename.

mgisymbols Unique gene symbols for TF.

tf_info Data frame, including extra information for all existing PWMs, it maybe different with motif dataset, default:NULL.

tf_missing Data frame, including extra information for missing PWMs, it maybe different with motif dataset, default:NULL.

distancematrix Distance matrix between motifs returned by `tfbs.clusterMotifs`, default:NULL.

expressionlevel Data frame indicatig the result of expression level returned by `tfbs.selectExpressedMotifs` or `tfbs.getExpression`, default:NULL.

cluster Matrix with 3 columns returned by `tfbs.clusterMotifs`, 1st column is the index of motifs, 2nd column is the group number of clustering, 3rd column is selected flag by the function `tfbs.selectByGeneExp` or `tfbs.selectByRandom`. default:NULL.

Methods

tfbs.importMotifs Import the licensed motifs or other missing motifs for tfbs object

tfbs.getExpression Estimate gene expression of target TF.

tfbs.selectExpressedMotifs Select the expressed motifs in GRO-seq, PRO-seq or RNA-seq experimental data.

tfbs.clusterMotifs Cluster the specified motifs and drawing the heatmap.

tfbs.scanTFsite Find TF sites from genome data within the BED ranges.

tfbs.enrichmentTest Comparative TFBS search with the BED ranges

tfbs.selectByGeneExp Select the motifs with minimum p-value from each group of clustering.

tfbs.selectByRandom Select the motifs randomly from each group of clustering.

tfbs.drawLogosForClusters Draw the motif logos by one group per page.

tfbs.drawLogo Draw the logo for a single TF motif.

See Also

The class definition of tfbs.

Examples

```
showClass("tfbs")
```

`tfbs.clusterMotifs` *Clustering the specified motifs and drawing the heatmap.*

Description

Clustering the specified motifs and drawing the heatmap.

Usage

```
tfbs.clusterMotifs(tfbs,
      method = c("agnes", "apcluster"),
      pdf.heatmap = NA,
      group.k = NA,
      apcluster.q = 0.99,
      ncores = 1,
      plot.style = c("rtfbsdb", "apcluster"),
      BG = log(c(0.25, 0.25, 0.25, 0.25)),
      ...)
```

Arguments

`tfbs`	A tfbs object (`"tfbs"`) returned by `tfbs.createFromCisBP`, `tfbs.dirs` or other functions.
`method`	String, availabe values are "agnes" and "apcluster".
`pdf.heatmap`	String, a PDF filename for heatmap.
`group.k`	Integer, if the method of agnes is used to do clustering, the parameter of k is optional to use as preset group number.
`apcluster.q`	Numeric value between 0 and 1, if the method of apcluster is used to do clustering, the parameter of q is optional to use as preset group number.
`ncores`	Number, the number of cores to use simultaneously.
`plot.style`	String indicating the heatmap is plotted by the apcluster package or not if the method apcluster is used.
`BG`	The log value of probabilities for nucleotide A, C, G and T as Backgroud computing.
`...`	The parameters used in function `apcluster`.

Details

This result of clustering will be used in the `tfbs.drawLogosForClusters`, `tfbs.selectByGeneExp`, `tfbs.enrichmentTest`.

tfbs@cluster will be updated by the clustering matrix which 1st column is the index of motifs and 2nd column is the group number of clustering.

Value

A matrix with 2 columns is returned, 1st column is the index of motifs and 2nd column is the group number of clustering.

See Also

See Also as tfbs.selectByGeneExp and tfbs.selectByRandom

Examples

```
# Load the internal CisBP data set
db <- CisBP.extdata("Homo_sapiens");

# Create a tfbs object by querying the meta file of CisBP dataset.
tfs <- tfbs.createFromCisBP(db, motif_type="ChIP-seq", tf.information.type=1 );

# Cluster the motifs using the "agnes" method
tfs <- tfbs.clusterMotifs(tfs, method="agnes",
      pdf.heatmap = "test-heatmap-agnes.pdf" );
show(tfs@cluster);

# Cluster the motifs using the "apcluster" method
tfs <- tfbs.clusterMotifs(tfs, method="apcluster",
      pdf.heatmap = "test-heatmap-apcluster.pdf" );
show(tfs@cluster);

# draw motif logos on one group per page.
tfbs.drawLogosForClusters(tfs, file.pdf = "test-cluster-logos.pdf");
```

---

tfbs.createFromCisBP *Create TF object by querying the CisBP dataset.*

Description

Build a tfbs object by querying the meta file of CisBP dataset and subsetting the results.

Usage

```
tfbs.createFromCisBP(cisbp.db,
      motif_id = NULL,
      tf_name = NULL,
      tf_status = NULL,
      family_name = NULL,
      motif_type = NULL,
      msource_id = NULL,
      tf.information.type = 1)
```

Arguments

cisbp.db	A CisBP object("CisBP.db"), including the file of TF_Information.txt.
motif_id	String, indicating the Motif_ID field will be used to select motifs.
tf_name	String, indicating the TF_name field will be used to select motifs.
tf_status	String, indicating the TF_Status field will be used to select motifs.
family_name	String, indicating the Family_Name field will be used to select motifs.

`motif_type` String, indicating the Motif_Type field will be used to select motifs.

`msource_id` String, indicating the MSource_Identifier field will be used to select motifs.

`tf.information.type`
Number,indicating which TF meta file will be used. Available values are 1 for TF_Information.txt, 2 for TF_Information_all_motifs.txt and 3 for TF_Information_all_motifs_plus.txt.

Details

The function includes three steps to build a tfbs object:
1) Searching the TF information and PWM files in the CisBP dataset according to the criteria specified by the parameters of *tf_name*, *tf_status*, *family_name*, *motif_type* and *msource_id*.

Value

A tfbs object is returned with PWM matrices, see Also as "tfbs"

See Also

See Also as tfbs

Examples

```
# Load the internal CisBP dataset
db_human <- CisBP.extdata("Homo_sapiens");

# Load all motifs and return a tfbs object.
tfs0 <- tfbs.createFromCisBP(db_human);

# Query the motifs by the conditions and return a tfbs object
tfs1 <- tfbs.createFromCisBP(db_human, family_name="Homeodomain", tf_status="D",
      motif_type="ChIP-seq", msource_id= "MS01_1.01", tf.information.type=1 );

# Query the motifs by the conditions and return a tfbs object
tfs2 <- tfbs.createFromCisBP(db_human, family_name="Homeodomain", tf_status="D" );

# Query the motifs by the conditions and return a tfbs object
tfs3 <- tfbs.createFromCisBP(db_human, motif_type="ChIP-seq"  );

# Query the motifs by the conditions and return a tfbs object
tfs4 <- tfbs.createFromCisBP(db_human, tf.information.type=2);
```

---

tfbs.createFromMotifDb

Create TF object by querying the MotifDb dataset.

Description

Create TF object from the MotifDb object which is a annotated collection of Protein-DNA binding sequenece motifs. The subset of MotifDb can be obtained by the methods in the MotifDb package or by the criteria in this function.

Usage

```
tfbs.createFromMotifDb(motifDB= NULL,
      organism = "Hsapiens",
      geneSymbol = NULL,
      tfFamily = NULL,
      providerName = NULL,
      providerId = NULL,
      dataSource = NULL,
      geneId = NULL,
      geneIdType = NULL,
      proteinId = NULL,
      proteinIdType = NULL,
      sequenceCount = NULL,
      bindingSequence = NULL,
      bindingDomain = NULL,
      experimentType = NULL,
      pubmedID = NULL,
      pseudocount = -7)
```

Arguments

motifDB	MotifDb object or subset of MotifDb.
organism	String, species, use command `print (table (values (MotifDb)$organism))` to check full list.
geneSymbol	String, gene symbol is used as Motif_ID in "tfbs", use command `print (table (values (MotifDb)$geneSymbol))` to check full list.
tfFamily	String, TF family is used as TF_Name in "tfbs", use command `print (table (values (MotifDb)$tfFamily))` to check full list.
providerName	String, use command `print (table (values (MotifDb)$providerName))` to check full list.
providerId	String, use command `print (table (values (MotifDb)$providerId))` to check full list.
dataSource	String, use command `print (table (values (MotifDb)$dataSource))` to check full list.
geneId	String, use command `print (table (values (MotifDb)$geneId))` to check full list.
geneIdType	String, use command `print (table (values (MotifDb)$geneIdType))` to check full list.
proteinId	String, use command `print (table (values (MotifDb)$proteinId))` to check full list.
proteinIdType	String, use command `print (table (values (MotifDb)$proteinIdType))` to check full list.
sequenceCount	String, use command `print (table (values (MotifDb)$sequenceCount))` to check full list.
bindingSequence	String, use command `print (table (values (MotifDb)$bindingSequence))` to check full list.

bindingDomain	String, use command `print (table (values (MotifDb)$bindingDomain))` to check full list.
experimentType	String, use command `print (table (values (MotifDb)$experimentType))` to check full list.
pubmedID	String, use command `print (table (values (MotifDb)$pubmedID))` to check full list.
pseudocount	Number, log value for zero value in PWM matrix, default is -7.

Details

Two methods to make a subset obeject.

1.Using the methods provided by the MotifDB package, please check the maunal of MotifDb package.

2. Searching the meta information and PWM matrices in the MotifDb object according to the criteria specified by the parameters of *organism*, *geneSymbol*, *tfFamily*, *providerName*, etc.

Value

A tfbs object is returned with PWM matrices, see Also as "tfbs"

Author(s)

MotifDB is authored by Paul Shannon.

References

The MotifDB is in Bioconductor. http://bioconductor.org/packages/release/bioc/html/MotifDb.html

See Also

See Also as "tfbs"

Examples

```
library(rtfbsdb);

if( require(MotifDb) )
{
# Load the subset of MotifDb generated by the method in 'MotifDb' package
mdb.sox4 <- query (MotifDb,'sox4');
tfs0 <- tfbs.createFromMotifDb(mdb.sox4, organism=NULL);

# Load the subset of MotifDb generated by the method in 'MotifDb' package
mdb.human <- query(MotifDb, 'hsapiens');
tfs1 <- tfbs.createFromMotifDb(mdb.human, organism=NULL);
}

# Load all motifs.
tfs2 <- tfbs.createFromMotifDb(organism=NULL);

show(MotifDb);
```

```
head(unique(tfs2@tf_info$organism), n=30);
head(unique(tfs2@tf_info$tfFamily), n=30);
head(unique(tfs2@tf_info$dataSource), n=30);
head(unique(tfs2@tf_info$experimentType), n=30);
head(unique(tfs2@tf_info$geneIdType), n=30);
head(unique(tfs2@tf_info$bindingDomain), n=30);

# Load all motifs of mouse species.
tfs4 <- tfbs.createFromMotifDb(organism="Mmusculus");

# Default: Load all motifs of human species.
tfs5 <- tfbs.createFromMotifDb();

# Load all motifs of Drosophila.melanogaster species.
tfs.Drosophila.melanogaster <- tfbs.createFromMotifDb(organism="Dmelanogaster");

# TFBS scanning
tfs.ap2 <- tfbs.createFromMotifDb(organism = "Hsapiens", tfFamily="AP2");

gen.bed <- data.frame(chr="chr19",
      start=round(runif(10,1000000, 2000000)),
      stop=0,
      name="",
      score=0,
      strand=".");
gen.bed$stop <- gen.bed$start + 3000;

file.twoBit <- system.file("extdata","hg19.chr19.2bit", package="rtfbsdb")

t1 <- tfbs.scanTFsite( tfs.ap2,
      file.twoBit,
      gen.bed,
      file.prefix="test.db",
      ncores = 1);
```

`tfbs.db-class` *Class* "`tfbs.db`"

Description

Abstract class for motif dataset. The CisBP class is a son class of tfbs.db.

Objects from the Class

Now code or function can be used to create this class.

Slots

`species`**:** Species name.

Methods

No methods defined with class "tfbs.db" in the signature.

See Also

"CisBP.db" inherits this class.

Examples

```
showClass("tfbs.db")
```

`tfbs.dirs` *Create a tfbs object from the folders.*

Description

Create a tfbs object from all the PWM files found in the supplied folders.

Usage

```
tfbs.dirs(...,
      species = "Homo_sapiens",
      args.read.motif = NULL,
      pattern = glob2rx("*.pwm"),
      recursive = FALSE)
```

Arguments

`...`	Multiple strings, one or more folders can be used in this function.
`species`	String, including the species name.
`args.read.motif`	List, including *pseudocount*, *force_even* or other parameters used in `read.table` function.
`pattern`	String, a character vector specifying regular expression and wlidcards.
`recursive`	Logical, indicating the loading recursively descends into subfolders or not, default: FALSE.

Details

Two parameters in the list of `args.read.motif` can be used:
pseudocount: log value for zero value in PWM matrix, default is -7.
force_even: whether the PWM matrix with odd size needs to be even.

Value

A tfbs object collecting all the PWM files in the specified folders. For the details of tfbs object, please see `tfbs`

See Also

The structure of tfbs object is described in "tfbs"

Examples

```
fs.dir <- system.file("extdata","", package="rtfbsdb")
tfs <- tfbs.dirs( fs.dir,
      args.read.motif = list(pseudocount=-7, header=TRUE, sep="\t" , row.names=1) );
str(tfs);
```

`tfbs.drawLogo` *Draw single motif logo.*

Description

Draw the motif logos in two models, 1 logo within a page or 1 group within one page.

Usage

```
tfbs.drawLogo(tfbs, file.pdf = NULL, index = NULL, tf_id = NULL,
                motif_id = NULL, tf_name = NULL, family_name = NULL,
                tf_status = NULL, groupby = NULL)
```

Arguments

`tfbs`	A tfbs object("tfbs")
`file.pdf`	String, the file name of PDF report.
`index`	Vector of number, indicating the motif index.
`tf_id`	Vector of string, indicating the TF_ID string, TF_ID is one motif attribute in TF_Information.txt. (Default=NULL).
`motif_id`	Vector of string, indicating the Motif_ID string, Motif_ID is one motif attribute in TF_Information.txt. (Default=NULL).
`tf_name`	Vector of string, indicating the TF_Name string, TF_Name is one motif attribute in TF_Information.txt. (Default=NULL).
`family_name`	Vector of string, indicating Family_Name string, Family_Name is one motif attribute in TF_Information.txt. (Default=NULL).
`tf_status`	String, indicating the TF_status value, TF_status is one motif attribute in TF_Information.txt. (Default=NULL).
`groupby`	String, indicating the group field is applied to print the motif, each group is printed in one page, the available values are NA, "Family_Name", "TF_Name", "TF_Status" or "Motif_Type". (Default=NA).

Details

Multiple selection is provided for outputting logos. The selected motifs by each criteria will be combined into one set.

Draw the motif logos in two models:
(1) 1 logo within a page (2) 1 group within one page.The motif logos are splitted if motif count is greater than 10.

Value

No return values.

See Also

See Also as "tfbs"

Examples

```
db <- CisBP.extdata("Homo_sapiens");

tfs <- tfbs.createFromCisBP(db);

motif_id   <- c( "M5604_1.01", "M5441_1.01", "M5162_1.01", "M5352_1.01");
tf_id      <- c( "T093250_1.01", "T093251_1.01","T093252_1.01","T093253_1.01");
family_name<- c( "p53", "Homeodomain", "Paired box", "Pipsqueak");


#Draw 10 motif logos from first one.
tfbs.drawLogo(tfs, file.pdf="test-drawLogo1.pdf", index=c(1:10) );

#Draw logos for specified Motif_ID, or TF_ID, or TF_Name, or Family_Name
tfbs.drawLogo(tfs, file.pdf="test-drawLogo2.pdf",
      motif_id = motif_id,
      tf_id = tf_id,
      tf_name = "AP-2",
      family_name = family_name,
      groupby = "TF_Status");

#Draw logos for specified TF_Status
tfbs.drawLogo(tfs, file.pdf="test-drawLogo3.pdf", tf_status="D",
      groupby="TF_Status");

#unlink("test-drawLogo1.pdf");
#unlink("test-drawLogo2.pdf");
#unlink("test-drawLogo3.pdf");
```

---

tfbs.drawLogosForClusters

Draw the motif logos by clustering.

Description

Draw the motif logos by one cluster per page.

Usage

```
tfbs.drawLogosForClusters(tfbs, file.pdf )
```

Arguments

tfbs	A tfbs object("tfbs").
file.pdf	String indicating a PDF eilname.

Details

It is different with tfbs.drawLogo which is capable of printing out motif logos in group. This group is calculated by the tfbs.clusterMotifs, not is classfied by any group filed.

Value

No return value.

See Also

See Also as tfbs.clusterMotifs

Examples

```
# Load the internal CisBP data set
db <- CisBP.extdata("Homo_sapiens");

# Create a tfbs object by querying the meta file of CisBP dataset.
tfs <- tfbs.createFromCisBP(db, motif_type="ChIP-seq", tf.information.type=1 );

# Cluster the motifs using the "cors" method
tfs <- tfbs.clusterMotifs(tfs, method="apcluster",
      pdf.heatmap = "test-heatmap1.pdf" );
show(tfs@cluster);

# draw motif logos on one group per page.
tfbs.drawLogosForClusters(tfs, "test-cluster1.pdf")
```

---

tfbs.enrichmentTest *Comparative TS sites between positive and negative TRE loci*

Description

Comparative TS sites between positive and negative TRE loci for all motifs.

Usage

```
tfbs.enrichmentTest(tfbs,
      file.twoBit,
      positive.bed,
      negative.bed=NA,
      file.prefix=NA,
      use.cluster=FALSE,
      ncores=1,
```

```
        gc.correction=TRUE,
        gc.correction.pdf=NA,
        gc.robust.rep=NA,
        threshold = 6,
        threshold.type = c("score", "fdr"),
        gc.groups=1,
        background.order=2,
        background.length=100000,
        pv.adj = p.adjust.methods)
```

Arguments

`tfbs`	A tfbs object, see also "`tfbs`"
`file.twoBit`	String, the file name of genome data(e.g. hg19.2bit, mm10.2bit)
`positive.bed`	Data frame, bed-formatted TRE loci.
`negative.bed`	Data frame, bed-formatted background loci. If not specified, the genomic loci adjacent to positive one are randomly extracted as the negative bed.
`file.prefix`	String, the prefix for outputted BED file, no bed files output if NA
`use.cluster`	Clustering matrix with 2 columns, 1st column is the index of motifs and 2nd column is the group number of clustering. It can be obtained from `tfbs.clusterMotifs`. If no clustering matrix, all motifs are used to do the comparson. see *details*
`ncores`	Number, comupting nodes in parallel environment.(default=1)
`gc.correction`	Logical value, if the difference between positive and negative TREs is significant,the resampling will be applied to the correction for the negative TREs. (default=TRUE)
`gc.correction.pdf`	String, indicating the pdf file name if the GC correction is checked. (default=NA)
`gc.robust.rep`	Number, indicating whether resampling background set multiple times is applied to get the median of binding sites. (default=NA)
`threshold`	Numeric value, if 'score' is specified in `threshold.type`, only binding sites with scores above this threshold are returned, if 'fdr' is specified in `threshold.type`, only binding sites with FDR (False Discovery Rate) less than this value can be selected. Default value is 6 for 'score' and 0.1 for 'fdr'.
`threshold.type`	String value, two options are available. only sites with scores above this threshold are returned, not be used if NA. (default = 'score')
`gc.groups`	Numeric value,indicating number of quantiles to group sequences into in `rtfbs` package. (default = 1)
`background.order`	Number, order of Markov model to build background.(default=2).
`background.length`	Number, length of the sequence to simulate background.(default=100000).
`pv.adj`	String, P-values correct method for `p.adjust` function. The available values are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr" or "none". (default="bonferroni").

Details

The difference of GC contents between positive.bed and negative.bed is checked before the comparson. The p-value of Wilcoxon-Mann-Whitney test shows this difference and helps the user to determine whether the GC correction is necessary.If the difference is very significant, please set `gc.correction` to do GC content correction by resampling the TREs from negative bed data based on the frequency of TREs in negative bed data. Use the parameter of `gc.correction.pdf` to output vioplot figurs in a pdf file if you want to check the visualized difference.

The clustering matix indicates which motifs in the 1st column are slected to do comparson and which clustering group in the 2nd columns are applied to adjust p-values for multiple comparsons. The function applys the p-values adjust for each clustering group. If no clustering information, all motifs in the tfbs object will be selected and adjusted as one group, which is the most conservative method.

Value

A object with the class name of "tfbs.enrichment" will be resturned in this comparson function. It includes one list of parameters `parm` and one data frame of results `result`.

`result` is a data frame with the following columns:

`motif.id`	Motif ID.
`tf.name`	TF name.
`Npos`	TF site count found in positive ranges.
`expected`	TF site count found in negative ranges.
`fe.ratio`	Ratio of fold enrichment.
`pvalue`	p-value calculated by fisher test.
`pv.adj`	p-value corrected by the multiple correction.
`starch`	Binary filename of detected TF sites.

The `result` can be outputted to a report by the function `tfbs.reportEnrichment`.

See Also

`print.tfbs.enrichment`, `summary.tfbs.enrichment`, `tfbs.reportEnrichment`.

Examples

```
library(rtfbsdb);

file.twoBit <- system.file("extdata","hg19.chr19.2bit", package="rtfbsdb")

db <- CisBP.extdata("Homo_sapiens");
tfs <- tfbs.createFromCisBP(db, family_name="AP-2");


#make two dummy BED data frame for positive loci and negative loci
pos.bed <- data.frame(chr="chr19",
      start=round(runif(1000,1000000, 2000000)),
      stop=0,
      name="",
```

```
      score=0,
      strand=".");
pos.bed$stop <- pos.bed$start + round(runif(1000, 20, 30));

neg.bed <- data.frame(chr="chr19",
      start=round(runif(8000, 800000, 1800000)),
      stop=0,
      name="",
      score=0,
      strand=".");
neg.bed$stop <- neg.bed$start + round(runif(8000, 20, 30));

t1 <- tfbs.enrichmentTest( tfs,
      file.twoBit,
      pos.bed,
      neg.bed,
      gc.correction=TRUE,
      ncores = 1); #ncores=3

#Show a brief result
t1;

#Show the comparson results of all motifs
show(t1$result);

summary(t1);

#Output the result to one pdf report.
tfbs.reportEnrichment(tfs, t1, file.pdf="test-tfbs-enrich-all.pdf", sig.only=FALSE);

file.ELF1 <- system.file("extdata","Chipseq-k562-chr19-ELF1.bed", package="rtfbsdb")
pos.bed<- read.table(file.ELF1)

tfs <- tfbs.createFromCisBP(db, family_name="Ets");

t2 <- tfbs.enrichmentTest( tfs,
      file.twoBit,
      pos.bed,
      neg.bed,
      gc.correction=TRUE,
      gc.robust.rep=5,
      ncores = 1); #ncores=3

show(t2)

#Output the result to one pdf report.
tfbs.reportEnrichment(tfs, t2, file.pdf="test-tfbs-enrich-both.pdf",
      sig.only=TRUE, enrichment.type="both");

t3 <- tfbs.enrichmentTest( tfs,
      file.twoBit,
      pos.bed,
      gc.correction=TRUE,
      gc.robust.rep=5,
      ncores = 1); #ncores=3

show(t3)
```

```
tfbs.reportEnrichment(tfs, t3, file.pdf="test-elf1-enrich-depleted.pdf",
      sig.only=TRUE, enrichment.type="depleted");
```

`tfbs.getExpression` *Estimate gene expression of target TF.*

Description

Gets expression level of target TF.
USE extra_info$DBID to find gene information encoded by GENCODE

Usage

```
tfbs.getExpression(tfbs,
      file.twoBit,
      file.gencode.gtf,
      file.bigwig.plus=NA,
      file.bigwig.minus=NA,
      file.bam=NA,
      seq.datatype = c("GRO-seq", "PRO-seq", "RNA-seq"),
      ncores =1 )
```

Arguments

`tfbs`	A tfbs object("`tfbs`").
`file.twoBit`	String, indicating the binary data of sequence. (e.g. hg19.2bit, mm10.2bit)
`file.gencode.gtf`	Gencode RDATA file encoded by ths package.
`file.bigwig.plus`	String, indicating bigwig file for strand plus(+) if `seq.datatype` is GRO-seq or PRO-seq.
`file.bigwig.minus`	String, indicating bigwig file for strand minus(-) if `seq.datatype` is GRO-seq or PRO-seq.
`file.bam`	String, indicating BAM file for rna reads if `seq.datatype` is RNA-seq.
`seq.datatype`	String,indicating which kind of seq data is applied to this function, three values are available: GRO-seq, PRO-seq and RNA-seq. (Default=GRO-seq)
`ncores`	Number, comupting nodes in parallel environment.

Details

For each motif, the occurance ranges can be queried by the gene ID in the GENCODE database(for human, gencode.v19.annotation.gtf, for mouse: gencode.vM3.annotation.gtf). After the searching, one range obtianed from the merge of the multiple ranges will be used to detect the reads count in the specified bigwig files(including plus and minus). The probability of each motif can be calcuated by the reads count and lambda.

The lambda is determined by the following formulation:

```
r.lambda = 0.04 * sum(reads_in_all_chromosomes)/10751533/1000.
```

The dataset of GENECODE v19 (human) and vM3 (mouse) have been compiled into RDATA file and attached in this package.

The `gencode_transcript_ext` object can be accessed after the following command is executed successfully.

```
load( system.file("extdata", "gencode_human21_transcript_ext.rdata",      package="rtfbsdb"), env
```

Value

A tbfs object with new expression data frame including the follwing columns:

`Motif_ID`	Motif_ID from CisBP dataset or other data source.
`DBID`	DBID from CisBP dataset or other data source.
`chr`	String, chromosome name.
`start`	Integer, start postion in which gene ID can be detected.
`end`	Integer, end postion in which gene ID can be detected.
`strand`	String, + or -, indicating the strand direction.
`bed_length`	Integer, the length of range which gene ID can be detected.
`reads`	The reads number queried by BigWig function from the bigwig files(plus and minus)
`lambda`	The lambda parameter in poison distribution.
`reads.RPKM`	The RPKM value of `reads` column.
`lambda.RPKM`	The RPKM value of `lambda` column.
`prob`	The probability calculated based on Poisson distribution.

See Also

See Also as "tfbs"

Examples

```
# Load the internal CisBP data set
db.human <- CisBP.extdata("Homo_sapiens");

# Create a tfbs object by querying the meta file of CisBP dataset.
tfs <- tfbs.createFromCisBP(db.human, motif_type="ChIP-seq",
      tf.information.type=1 );

file.bigwig.minus <- system.file("extdata","GSM1480327_K562_PROseq_chr19_minus.bw",
      package="rtfbsdb")
file.bigwig.plus <- system.file("extdata","GSM1480327_K562_PROseq_chr19_plus.bw",
      package="rtfbsdb")
hg19.twobit <- system.file("extdata","hg19.chr19.2bit", package="rtfbsdb")
```

```
gencode.gtf <- system.file("extdata","gencode.v19.annotation.chr19.gtf.gz",
      package="rtfbsdb")

tfs <- tfbs.getExpression(tfs, hg19.twobit, gencode.gtf,
      file.bigwig.plus, file.bigwig.minus, ncore=1);
```

`tfbs.importMotifs` *Import motifs to tfbs object*

Description

Import licensed motifs or the source other than Cis-BP to tfbs object

Usage

```
tfbs.importMotifs(tfbs, format, filenames, motif_ids=NULL, PPM.format=TRUE,
      skip.lines=0, pseudocount= -7, force_even= FALSE, ...)
```

Arguments

`tfbs`	A tfbs object ("`tfbs`") returned by `tfbs.createFromCisBP`, `tfbs`, `tfbs.dirs`.
`format`	String value indicating predefined format or string vector containing some tags to define a customize format. Predefined formats include **pwm.matrix**, **jaspar**, **meme**, **mscan**, **transfac** and **HOCOMOCO**.
`filenames`	Vector of file names.
`motif_ids`	Vector of motif ID only for the format **pwm.matrix** . Theses motif IDs are assigned to the motif data loaded from 'pwm.matrix' files. If these motifs are missing or licensed motifs, these IDs have to be in accordance with the TF information which can be exported from 'TF_information.txt' by the function `CisBP.getTFinformation`.
`PPM.format`	Boolean value indicating whether the matrix data is format by PPM(position probability matrix) or PFM(position frequency matrix), otherwise, the original data is PWM format. Default: FALSE.
`skip.lines`	Number indicating specified non-empty lines are skipped from the head of data file. This parameter is not used for the format 'pwm.matrix'. Default: 0.
`pseudocount`	Number value indicating to replace -Inf in PWM. Default: -7.
`force_even`	Boolean value indicating whether to append a row to make the row number even. Default: FALSE.
`...`	Optional paramaters used in the function `read.table` for the 'pwm.matrix' format.

Details

Two goals to import the motifs from the sources other than Cis-BP.
1) Fill the missing motifs mainly licensed in the Cis-BP database.
2) Make use of the different source.

Five predefined formats are available, **'pwm.matrix'**, **'jaspar'**, **'meme'**, **'mscan'**, **'transfac'** and **'HOCOMOCO'**.

'pwm.matrix'
Single text file containing 5 colums, Position, A, C, G and T as shown below.

```
Pos A C G T
1 0.2 0.0 0.4 0.4
2 0.0 0.4 0.4 0.2
3 0.2 0.2 0.2 0.4
4 0.6 0.0 0.4 0.0
5 0.0 0.4 0.0 0.6
6 0.0 0.4 0.4 0.2
7 0.0 0.0 1.0 0.0
8 0.2 0.0 0.0 0.8
9 0.0 0.6 0.0 0.4
10 0.6 0.4 0.0 0.0
11 0.0 1.0 0.0 0.0
```

'jaspar'
The package use a predifined template to load the 'jaspar' format as show below.

```
> Mycn
A [ 0 29 0 2 0 0 ]
C [31 0 30 1 3 0 ]
G [ 0 0 0 28 0 31]
T [ 0 2 1 0 28 0 ]
```

'meme'
The 'meme' format reference: http://meme-suite.org/
The package use a predifined template to load the 'meme' format as show below.

```
MOTIF crp alternative name
letter-probability matrix: alength= 4 w= 19 nsites= 17 E= 4.1e-009
 0.000000  0.176471  0.000000  0.823529
 0.000000  0.058824  0.647059  0.294118
 0.000000  0.058824  0.000000  0.941176
 0.176471  0.000000  0.764706  0.058824
 0.823529  0.058824  0.000000  0.117647
 0.294118  0.176471  0.176471  0.352941
 0.294118  0.352941  0.235294  0.117647
 0.117647  0.235294  0.352941  0.294118
 0.529412  0.000000  0.176471  0.294118
 0.058824  0.235294  0.588235  0.117647
 0.176471  0.235294  0.294118  0.294118
 0.000000  0.058824  0.117647  0.823529
 0.058824  0.882353  0.000000  0.058824
 0.764706  0.000000  0.176471  0.058824
 0.058824  0.882353  0.000000  0.058824
```

```
 0.823529  0.058824  0.058824  0.058824
 0.176471  0.411765  0.058824  0.352941
 0.411765  0.000000  0.000000  0.588235
 0.352941  0.058824  0.000000  0.588235
```

'mscan'
The 'mscan' format reference: http://www.cisreg.ca/cgi-bin/mscan/MSCAN
The package use a predifined template to load the 'mscan' format as show below.

```
>mef2
10  0  0  0 22  0  6  2  3  4 22 10
0  2 12  0  0  0  0  0  0  0  0  0
9 20  2  0  0  0  0  0  0  0  0 10
3  0  8 22  0 22 16 20 19 18  0  2
>myf
7  9  4  0 16  7  0  6  0  0  6  0
8  0  2 15  0  0 15  0  0 10  0  0
1  7 10  1  0  9  1  0 16  6  0 16
0  0  0  0  0  0  0 10  0  0 10  0
```

'transfac'
The transfac format reference: http://www.cisreg.ca/cgi-bin/mscan/MSCAN
The package use a predifined template to load the 'transfac' format as show below.

```
AC  MA0001.1
XX
ID  AGL3
XX
DE  MA0001.1 AGL3; from JASPAR
PO       A      C      G      T
1        0     94      1      2
2        3     75      0     19
3       79      4      3     11
4       40      3      4     50
5       66      1      1     29
6       48      2      0     47
7       65      5      5     22
8       11      2      3     81
9       65      3     28      1
10       0      3     88      6
XX
CC  program: jaspar
XX
//
```

The package implemented a simple parser to load motifs from the different sources. This parser basically reads the data file word by word according to the format rules defined in advance for diffierent source.
The format rules are comprised of some fixed vocabularies and tags defined by the parser or the user. So there are two kinds of tags to describe a motif format, predefined tags and user-defined

tags. All tags start with the dollar($) symbol and meet the requirement of program identifier, such as $Express_Pvalue,$D1.
The predefined tags include the following names, which define the motif information and control the parser's cursor.

[1]	'>' '[' ']'	control	Start-stop characters for a line or a motif
[2]	$SKIP n	control	Ignore or skip the rest part or next *n* lines.
[3]	$REPEAT	control	Repeat use the current format until it can't be matched.
[4]	$EOM	control	End of Motif.
[5]	$LOM	control	Line of Motif.
[6]	$Motif_ID	variable	Motif_ID required in the 'tfbs' object.
[7]	$TF_Name	variable	TF_Name required in the 'tfbs' object.
[8]	$A+	variable	Multiple *A* nucleobase frequencies in one row
[9]	$C+	variable	Multiple *C* nucleobase frequencies in one row
[10]	$G+	variable	Multiple *G* nucleobase frequencies in one row
[11]	$T+	variable	Multiple *T* nucleobase frequencies in one row
[12]	$A	variable	Single *A* nucleobase frequence in one column
[13]	$C	variable	Single *C* nucleobase frequence in one column
[14]	$G	variable	Single *G* nucleobase frequence in one column
[15]	$T	variable	Single *T* nucleobase frequence in one column

The user-defined tags reprsent the variables which values can be collected to the 'tf_info' slot in the tfbs object ("tfbs"). e.g. $Description, $anyword.

The following section shows 5 predefined format:

'jaspar'

```
>$Motif_ID $TF_Name
A [ $A+ ]
C [ $C+ ]
G [ $G+ ]
T [ $T+ ]
```

'meme'

```
MOTIF $Motif_ID $TF_Name $SKIP
letter-probability matrix: $SKIP
$A  $C  $G  $T $REPEAT
URL $SKIP
```

'mscan'

```
>$Motif_ID $TF_Name $SKIP
$A+
$C+
$G+
$T+
```

'transfac'

```
AC  $Motif_ID
XX  $SKIP
ID  $TF_Name
XX  $SKIP
DE  $Description
PO       A     C     G     T
$LOM    $A    $C    $G     $T $REPEAT
XX $SKIP
CC $SKIP $REPEAT
// $EOM
```

'HOCOMOCO'

```
>$Motif_ID,
$A $C $G $T $REPEAT
```

Value

A new tfbs object ("tfbs") merged with licensed motifs.

References

1.http://rsat01.biologie.ens.fr/rsa-tools/help.convert-matrix.html
2.http://www.cisreg.ca/cgi-bin/mscan/MSCAN
3.http://meme-suite.org/
4.http://hocomoco.autosome.ru/downloads

See Also

tfbs.createFromCisBP

Examples

```
require(rtfbsdb);

db  <- CisBP.extdata("Homo_sapiens");
tfs <- tfbs.createFromCisBP(db, family_name="AP-2");
tfs;


#It is test code
#motif_ids <- c( "M2938_1.02", "M2940_1.02", "M2940_2.02", "M4056_1.02" );
#path <- system.file("extdata", package="rtfbsdb");
#file_pwms <- paste(path, c(
#      "fake_M2938_1.02.pwm",
#      "fake_M2940_1.02.pwm",
#      "M3591_1.01.pwm",
#      "M3590_1.01.pwm",
#      ), sep="/");
#
#
#tfs <- tfbs.importMotifs(tfs, 'pwm.matrix', file_pwms, motif_ids, header=T );
#show(tfs);
```

```
## import 2 motifs to fill the licensed motifs in Cis-BP and 1 new motif fromother source
motif_ids <- c( "M2938_1.02", "M3591_1.01", "M3590_1.01" );
path <- system.file("extdata", package="rtfbsdb");
file_pwms <- paste(path, c(
      "fake_M2938_1.02.pwm",
      "M3591_1.01.pwm",
      "M3590_1.01.pwm"), sep="/");

tfs <- tfbs.importMotifs(tfs, 'pwm.matrix', file_pwms, motif_ids, header=TRUE );
show(tfs);


## Data is copied from http://rsat01.biologie.ens.fr/rsa-tools/help.convert-matrix.html
##
data.transfac <- system.file("extdata", "pwm.example.transfac.txt", package="rtfbsdb");
tfs.transfac <- tfbs.importMotifs(tfs, "transfac", data.transfac, skip.lines=0);
show(tfs.transfac);

## Data from
## http://jaspar.genereg.net/html/DOWNLOAD/JASPAR_CORE/pfm/nonredundant/pfm_all.txt
##
data.jaspar <- system.file("extdata", "pwm.example.jaspar.2015.txt", package="rtfbsdb");
tfs.jaspar <- tfbs.importMotifs(tfs, "jaspar", data.jaspar, skip.lines=0);
show(tfs.jaspar);

## Data from
## http://jaspar.genereg.net/html/DOWNLOAD/ARCHIVE/JASPAR2010/JASPAR_CORE/pfms/pfms_all.txt
##
data.mscan <- system.file("extdata", "pwm.example.mscan.jaspar2010.txt", package="rtfbsdb");
tbs <- tfbs();
tfs.mscan <- tfbs.importMotifs(tbs, "mscan", data.mscan, skip.lines=0);
show(tfs.mscan);

## Data from http://meme-suite.org/doc/download.html?man_type=web
##
data.meme <- system.file("extdata", "pwm.example.meme.Homo_sapiens.txt", package="rtfbsdb");
tbs <- tfbs();
tfs.meme <- tfbs.importMotifs(tbs, "meme", data.meme, skip.lines=5);
show(tfs.meme);

## Data from http://www.nature.com/nature/journal/v527/n7578/full/nature15518.html
##
format.style <- c("Base $Motif_ID $TF_Name $Experiment
$Ligand_sequbatch $Seed $Multinomial $Cycle $Is_matrix $Comment
$Genomic_pvalue $Enrichment_pvalue $Category $SKIP",
"A $A+",
"C $C+",
"G $G+",
"T $T+" );
data.file <- system.file("extdata", "pwm.example.nature15518.s1.txt", package="rtfbsdb");

tbs <- tfbs();
tfs.nature15518 <- tfbs.importMotifs(tbs, format.style, data.file, skip.lines=19)
show(tfs.nature15518);
```

```
## HOCOMOCO_10
## Data from http://hocomoco.autosome.ru/downloads
## The matrix is PWM format.

library(rtfbsdb)
format.style <- c(">$Motif_ID",
"$A $C $G $T $REPEAT");
data.file <- system.file("extdata", "HOCOMOCO_10_pwms_HUMAN_mono.txt", package="rtfbsdb");

tbs <- tfbs();
tfs.HOCOMOCO.human <- tfbs.importMotifs(tbs, format.style, data.file, skip.lines=0, PPM.format=FALSE)
show(tfs.HOCOMOCO.human);
```

`tfbs.reportEnrichment` *Output report for enrichment results.*

Description

Output enrichment results to a PDF report which includes motif names, counts of TF site, p-value, enrichment ratio and motif logos.

Usage

```
tfbs.reportEnrichment(tfbs, r.comp,
      file.pdf = NA,
      report.size = "letter",
      report.title = "",
      enrichment.type = c ("both", "enriched", "depleted"),
      sig.only = TRUE,
      pv.threshold = 0.05,
      pv.adj = NA,
      sorted = c ("pvalue", "enrich.ratio"))
```

Arguments

`tfbs`	A tfbs object, see also "`tfbs`"
`r.comp`	A result object from the function of `tfbs.enrichmentTest`
`file.pdf`	String, the file name of PDF report.
`report.size`	String, the page size (default="letter")
`report.title`	String, the report title.
`enrichment.type`	String, three values are available for significant motifs to be printed out.(default="both").
`sig.only`	String, indicating whether only significant motifs are outputted or not.(default=TRUE).
`pv.threshold`	Numeric value,indicating whether the different threshold of p-value is applied to select the significant motifs.
`pv.adj`	String,indicating whether the different correction metod of p-value is applied to select the significant motifs.
`sorted`	String,indicating which field is used to sort the results and print in the report. (default="pvalue")

Details

The table with 7 columns is outputted into a PDF report within letter size.
Two color bars are used to display p-values and enrichment ratios. Motif logos are shown visually in each row.

Value

No return values.

See Also

tfbs.enrichmentTest, summary.tfbs.enrichment.

Examples

```
# see examples in tfbs.enrichmentTest
```

---

tfbs.reportFinding *Make report for scanning results.*

Description

Output a PDF report includes motif names, counts of TF site and motif logos.

Usage

```
tfbs.reportFinding(tfbs,
      r.scan,
      file.pdf = NA,
      report.size = "letter",
      report.title = "")
```

Arguments

tfbs	A tfbs object, see also "tfbs"
r.scan	A result object from the function of tfbs.scanTFsite
file.pdf	String, the file name of PDF report.
report.size	String, the page size (default="letter")
report.title	String, the report title.

Details

The table with 4 columns is outputted into a PDF report within letter size.
Motif logos are shown visually in each row.

Value

No return values.

See Also

tfbs.scanTFsite, print.tfbs.finding

Examples

```
#See example in tfbs.scanTFsite
```

---

tfbs.scanTFsite *Find TF sites from genome data within the BED loci*

Description

Find TF sites from genome data within the BED loci. Please notice that this package does not provided genome data such as hg19.2bit, mm10.2bit.

Usage

```
tfbs.scanTFsite(tfbs,
      file.twoBit,
      gen.bed,
     return.type=c("matches", "maxscore", "posteriors", "maxposterior", "writedb"),
      file.prefix=NA,
      usemotifs = NA,
      ncores = 1,
      threshold = 6,
      threshold.type = c("score", "fdr"),
      gc.groups = NA,
      background.order = 2,
      background.length = 100000)
```

Arguments

tfbs	A tfbs object ("tfbs") returned by tfbs.createFromCisBP, tfbs, tfbs.dirs.
file.twoBit	String, the file name of genome data(e.g. hg19.2bit or mm10.2bit)
gen.bed	Data frame, bed-formatted loci information with 6 columns
return.type	String, five available values explained in th details(default = "matches")
file.prefix	String, the prefix for outputted file, only used when the return.type is *writedb*
usemotifs	Vector indicating indexes of motif to be used in scanning.
ncores	Number, computing nodes in parallel environment (default = 1).
threshold	Numeric value, if 'score' is specified in threshold.type, only binding sites with scores above this threshold are returned, if 'fdr' is specified in threshold.type, only binding sites with FDR (False Discovery Rate) less than this value can be selected. Default value is 6 for 'score' and 0.1 for 'fdr'.
threshold.type	String value, two options are available. only sites with scores above this threshold are returned, not be used if NA. (default = 'score')
gc.groups	Numeric value,indicating number of quantiles to group sequences into in rtfbs package (default = 1).

`background.order`
: Numeric value,indicating the order of Markov model to build in `rtfbs` package (default = 2).

`background.length`
: Numeric value, indicating length of the sequence to simulate in `rtfbs` package (default = 100000)

Details

(1) Five options are availabe for the function of tfbs.scanTFsite as follows.

- matches: returns all matching TF sites for all motifs.
- writedb: writes a bed file with matches sites. Assumes that sort-bed and starch tools are availiable in $PATH
- maxscore: returns the max score(posterior difference between motif model and background) in each bed-formatted loci.
- posteriors: returns the posteriors at each position in bed-formatted loci.
- maxposterior: returns the max(posterior) in each bed-formatted loci.

(2) In order to make the binary file with the parameter of writedb, make sure that starchcat and sort-bed command (in BEDOPS) can be accessed from R environment. If not, please put the folder in $PATH.

Value

A list object will be returned with the class name of `tfbs.finding`. The object wraps four sub-list as follows:

1) `parm`:Calling parameters(`fdr`, `threshold`), `gc.groups`...).
2) `bed`: Calling bed-formatted loci(`gen.bed`).
3) `summary`: A data frame including summrized information about matched TF sites for all motifs.
4) `result`: Scanning results which data type is depend on the parameter of `return.type`.

The option of *matches* returns a list including the result of every motif, which result is BED style data frame with the following columns.

`chrom`	Chromosome
`chromStart`	Start position
`chromEnd`	Chromosome end position
`name`	
`score`	The score is given by the log likelihood ratio against the Marklov model(backgound).
`strand`	Strand
`peakStart`	Start position of the matched BED region
`peakEnd`	End position of the matched BED region

The option of *writedb* will return a binary BED filename in which store all bed ranges.

The option of *posteriors* will return a list for each motif returned by score.ms function. Scores represent the motif 'match score', or the product of the probability of observing each base under the motif or background models. Scores are returned under the motif model for all positions in the sequence, on both forward and reverse strands, and under the background model.

The option of *maxposterior* will return a probability matrix which the row indicates the target loci and the column indicates the motif.

The option of *maxscore* will return a score (poserior difference) matrix which the row indicates the target loci and the column indicates the motif.

See Also

print.tfbs.finding, summary.tfbs.finding, tfbs.reportFinding.

Examples

```
library(rtfbsdb);

file.twoBit <- system.file("extdata","hg19.chr19.2bit", package="rtfbsdb")

db <- CisBP.extdata("Homo_sapiens");
tfs <- tfbs.createFromCisBP(db, family_name="Ets");

gen.bed <- data.frame(chr="chr19",
      start=round(runif(10,1000000, 2000000)),
      stop=0,
      name="",
      score=0,
      strand=".");
gen.bed$stop <- gen.bed$start + 3000;

t1 <- tfbs.scanTFsite( tfs,
      file.twoBit,
      gen.bed,
      file.prefix="test.db",
      ncores = 1);

#show a brief information about the result
t1

#show the summary information in the result
show(t1$summary);

#show the matched TF sites for first motif
show(t1$result[[1]]);

#Output a PDF report for all motifs.
tfbs.reportFinding(tfs, t1, file.pdf="test-rtfbs-scan.pdf", report.title="ELF1");

file.ELF1 <- system.file("extdata","Chipseq-k562-chr19-ELF1.bed", package="rtfbsdb")
gen.bed<- read.table(file.ELF1)
```

```
t2 <- tfbs.scanTFsite( tfs,
      file.twoBit,
      gen.bed,
      file.prefix="test.db",
      return.type="writedb",
      ncores = 1);

t2

t3 <- tfbs.scanTFsite( tfs,
      file.twoBit,
      gen.bed,
      return.type="posteriors",
      ncores = 1);

t3

t4 <- tfbs.scanTFsite( tfs,
      file.twoBit,
      gen.bed,
      return.type="maxposterior",
      ncores = 1);

t4;

dim(t4$result);

t5 <- tfbs.scanTFsite( tfs,
      file.twoBit,
      gen.bed,
      return.type="maxscore",
      ncores = 1);

t5;
```

`tfbs.selectByGeneExp` *Motif selection by gene expression level.*

Description

Select the motifs with minimum p-value from each group of clustering.

Usage

```
tfbs.selectByGeneExp(tfbs)
```

Arguments

`tfbs`	A tfbs object ("`tfbs`") with the data frame of gene expression level.

Details

The function of tfbs.getExpression should be successfully called and the results of gene expression should be returned before this function is called. The indexes of selected motifs will be used in the function of tfbs.enrichmentTest or tfbs.scanTFsite.

Value

New tfbs object with the selected indices is returned(tfbs@cluster). The 3rd column of tfbs@cluster is added or updated as the select flag(1:selected, 0:unselected).

See Also

See Also as tfbs.selectByRandom, tfbs.getExpression

Examples

```
db <- CisBP.extdata("Homo_sapiens");

tfs <- tfbs.createFromCisBP(db, family_name="AP-2");


file.bigwig.minus <- system.file("extdata",
      "GSM1480327_K562_PROseq_chr19_minus.bw", package="rtfbsdb")
file.bigwig.plus <- system.file("extdata",
      "GSM1480327_K562_PROseq_chr19_plus.bw", package="rtfbsdb")
hg19.twobit <- system.file("extdata","hg19.chr19.2bit", package="rtfbsdb")
gencode.gtf <- system.file("extdata",
      "gencode.v19.annotation.chr19.gtf.gz", package="rtfbsdb")

tfs <- tfbs.getExpression(tfs, hg19.twobit, gencode.gtf,
      file.bigwig.plus, file.bigwig.minus, ncore=1);

tfs <- tfbs.clusterMotifs(tfs, pdf.heatmap="test-AP2-heatmap.pdf" );

tfs <- tfbs.selectByGeneExp( tfs );

show(tfs@cluster);
```

---

tfbs.selectByRandom *Random motif selection*

Description

Select the motifs randomly from each group of clustering.

Usage

```
tfbs.selectByRandom(tfbs)
```

Arguments

tfbs	A tfbs object("tfbs").

Details

The indexes of selected motifs can be used in the function of `tfbs.enrichmentTest` or `tfbs.scanTFsite`.

Value

New tfbs object with the selected indices is returned(tfbs@cluster). The 3rd column of tfbs@cluster is added or updated as the select flag(1:selected, 0:unselected).

See Also

See Also as `tfbs.selectByGeneExp`, `tfbs.getExpression`

Examples

```
db <- CisBP.extdata("Homo_sapiens");

tfs <- tfbs.createFromCisBP(db, family_name="AP-2");

tfs <- tfbs.clusterMotifs(tfs, pdf.heatmap="test-AP2-heatmap.pdf" );

tfs <- tfbs.selectByRandom(tfs );

show(tfs@cluster);
```

---

tfbs.selectExpressedMotifs

Select expressed Motifs for GRO-seq, PRO-seq and RNA-seq data

---

Description

Select expressed Motifs for GRO-seq, PRO-seq and RNA-seq data

Usage

```
tfbs.selectExpressedMotifs(tfbs,
      file.twoBit,
      file.gencode.gtf,
      file.bigwig.plus=NA,
      file.bigwig.minus=NA,
      file.bam=NA,
      seq.datatype= c("GRO-seq", "PRO-seq", "RNA-seq"),
      pvalue.threshold = 0.05,
      include.DBID.missing=TRUE,
      ncores = 1 )
```

Arguments

tfbs	A tfbs object ("tfbs") returned by tfbs.createFromCisBP, tfbs, tfbs.dirs.
file.bigwig.plus	String, indicating bigwig file for strand plus(+) if seq.datatype is GRO-seq or PRO-seq.
file.bigwig.minus	String, indicating bigwig file for strand minus(-) if seq.datatype is GRO-seq or PRO-seq.
file.bam	String, indicating BAM file for rna reads if seq.datatype is RNA-seq.
file.twoBit	String, indicating the binary data of sequence. (e.g. hg19.2bit, mm10.2bit)
file.gencode.gtf	String,indicating Gencode GTF file downloaded from the Gencode web site.
seq.datatype	String,indicating which kind of seq data is applied to this function, three values are available: GRO-seq, PRO-seq and RNA-seq. Default: GRO-seq
pvalue.threshold	Numeric, indicating .
include.DBID.missing	Logical, indicating whether the TFs without association with GENCODE through the DBID are selected.
ncores	Number, comuputing nodes in parallel environment for gencode data converting.

Details

1) If seq.datatype is GRO-seq or PRO-seq and the bigwig files are provided, the gene expression values are calculated through querying the TREs region from the GENCODE database(for human, gencode.v19.annotation.gtf, for mouse: gencode.vM3.annotation.gtf)
and querying the reads count in the plus and minus bigWig files.

If seq.datatype is RNA-seq and the BAM file is provided, read counts for each TRE regions will be queried from the BAM file.

2) If the expressed TFs only is used in the tfbs object, the TFs with p-values corrected by Bonfrroni less than 0.05 will be selected.

The following part explains how to calculate the gene expression.

For each motif, the occurance ranges can be queried by the gene ID After the searching, one range obtained from the merge of the multiple ranges will be used to detect the reads count in the specified bigwig files(including plus and minus). The probability of each motif can be calcuated by the reads count and lambda.

The lambda is determined by the following formulation:

For GRO-seq and PRO-seq data:
`r.lambda = 0.04 * sum(reads_in_all_chromosomes)/10751533/1000.`

For RNA-seq data:

```
r.lambda = mode( reads_in_1000_bp_windows_cross_all_gene_deserts )/1000.
```

This function will be failed to get the reads count if the BAM file is not indexed. Please use the command samtools to make the index file for the BAM file

```
samtools index your_bam_file
```

Value

A new tfbs object ("tfbs") with the matrix of gene expression level.

Examples

```
library(rtfbsdb);

# Load the internal CisBP data set
db.human <- CisBP.extdata("Homo_sapiens");

# Create a tfbs object by querying the meta file of CisBP dataset.
tfs <- tfbs.createFromCisBP(db.human, motif_type="ChIP-seq",
      tf.information.type=1 );

file.bigwig.minus <- system.file("extdata",
      "GSM1480327_K562_PROseq_chr19_minus.bw", package="rtfbsdb")
file.bigwig.plus <- system.file("extdata",
      "GSM1480327_K562_PROseq_chr19_plus.bw", package="rtfbsdb")
hg19.twobit <- system.file("extdata","hg19.chr19.2bit", package="rtfbsdb")
gencode.gtf <- system.file("extdata",
      "gencode.v19.annotation.chr19.gtf.gz", package="rtfbsdb")

tfs1 <- tfbs.selectExpressedMotifs(tfs,
      hg19.twobit,
      gencode.gtf,
      file.bigwig.plus,
      file.bigwig.minus,
      seq.datatype = "PRO-seq",
      pvalue.threshold=0.001,
      include.DBID.missing=TRUE,
      ncore=1);

show(tfs1)

file.bam <- "/local/storage/projects/NHP/AllData/bams/H3_U.fastq.gz.sort.bam"

tfs2 <- tfbs.selectExpressedMotifs(tfs,
      hg19.twobit,
      gencode.gtf,
      file.bam = file.bam,
      seq.datatype = "RNA-seq",
      pvalue.threshold=0.01,
      include.DBID.missing=TRUE,
      ncore=1);

show(tfs2)
```

Index