

The *TFBSTools* package overview

Ge Tan *

Edited: May 2014; Compiled: October 13, 2014

Contents

1	Introduction	1
2	S4 classes in TFBSTools	2
2.1	XMatrix and its subclasses	2
2.2	XMatrixList and its subclasses	5
2.3	SiteSet, SiteSetList, SitePairSet and SitePairSetList	6
2.4	MotifSet	6
3	Database interfaces for JASPAR2014 database	6
3.1	Search JASPAR2014 database	6
3.2	Store, delete and initialize JASPAR2014 database	6
4	PFM, PWM and ICM methods	7
4.1	PFM to PWM	7
4.2	PFM to ICM	8
4.3	Align PFM to a custom matrix or IUPAC string	9
4.4	PFM permutation	10
4.5	PWM similarity	11
5	Scan sequence and alignments with PWM pattern	12
5.1	searchSeq	12
5.2	searchAln	13
5.3	searchPairBSgenome	16
6	Use external pattern generators	16
6.1	MEME	16
7	Conclusion	16

1 Introduction

Eukaryotic regulatory regions are characterized based a set of discovered transcription factor binding sites, which can be represented as sequence patterns with various degree of degeneracy.

This *TFBSTools* package is designed to be a computational framework for transcription factor binding site analysis. Based on the famous perl module TFBS ([Lenhard and Wasserman, 2002](#)), we extended the class definitions and enhanced implementations in an interactive environment. So far this package contains a set of integrated R S4 style classes, tools

*ge.tan09@imperial.ac.uk

, JASPAR database interface functions. Most approaches can be described in three sequential phases. First, a pattern is generated for a set of target sequences known to be bound by a specific transcription factor. Second, a set of DNA sequences are analyzed to determine the locations of sequences consistent with the described binding pattern. Finally, in advanced cases, predictive statistical models of regulatory regions are constructed based on multiple occurrences of the detected patterns.

TFBSTools aims to support all these functionalities in the environment *RR*, except the external motif finding software, such as *MEME* (Bailey and Elkan, 1994).

2 S4 classes in TFBSTools

The package is built around a number of S4 class of which the *XMatrix*, *SiteSet* classes are the most important. The section will briefly explain most of them defined in *TFBSTools*.

2.1 XMatrix and its subclasses

XMatrix is a virtual class, which means no concrete objects can be created directly from it. The subclass *PfMatrix* is designed to store all the relevant information for one raw position frequency matrix (PFM). This object is compatible with one record from JASPAR database.

PWMMatrix is used to store a position weight matrix (PWM). Compared with *PfMatrix*, it has one extra slot pseudo-counts.

ICMatrix is used to store a information content matrix (ICM). Compared with *PWMMatrix*, it has one extra slot *schneider*.

```
> library(TFBSTools)
> pfm = PfMatrix(ID="MA0004.1", name="Arnt", matrixClass="Zipper-Type", strand="+",
+             bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
+             tags=list(family="Helix-Loop-Helix", species="10090",
+                       tax_group="vertebrates", medline="7592839", type="SELEX",
+                       ACC="P53762", pazar_tf_id="TF0000003",
+                       TFBSshape_ID="11", TFencyclopedia_ID="580"),
+             profileMatrix=matrix(c(4L, 19L, 0L, 0L, 0L, 0L,
+                                   16L, 0L, 20L, 0L, 0L, 0L,
+                                   0L, 1L, 0L, 20L, 0L, 20L,
+                                   0L, 0L, 0L, 0L, 20L, 0L),
+                                 byrow=TRUE, nrow=4,
+                                 dimnames=list(c("A", "C", "G", "T"))))
+             )
> ## coerced to matrix
> as.matrix(pfm)
  [,1] [,2] [,3] [,4] [,5] [,6]
A    4   19    0    0    0    0
C   16    0   20    0    0    0
G    0    1    0   20    0   20
T    0    0    0    0   20    0

> ## access the slots of pfm
> ID(pfm)
[1] "MA0004.1"

> name(pfm)
[1] "Arnt"
```

```

> Matrix(pfm)
  [,1] [,2] [,3] [,4] [,5] [,6]
A    4   19    0    0    0    0
C   16    0   20    0    0    0
G    0    1    0   20    0   20
T    0    0    0    0   20    0

> ## conversion to pwm, icm
> pwm = toPWM(pfm)
> pwm

An object of class PWMMatrix
ID: MA0004.1
Name: Arnt
Matrix Class: Zipper-Type
strand: +
Pseudocounts: 0.8
Tags:
$family
[1] "Helix-Loop-Helix"

$species
[1] "10090"

$tax_group
[1] "vertebrates"

$medline
[1] "7592839"

$type
[1] "SELEX"

$ACC
[1] "P53762"

$pazar_tf_id
[1] "TF0000003"

$TFBSshape_ID
[1] "11"

$TFencyclopedia_ID
[1] "580"

Background:
  A    C    G    T
0.25 0.25 0.25 0.25
Matrix:
  [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
A -0.3081223  1.884523 -4.700440 -4.700440 -4.700440 -4.700440
C  1.6394103 -4.700440  1.957772 -4.700440 -4.700440 -4.700440
G -4.7004397 -2.115477 -4.700440  1.957772 -4.700440  1.957772
T -4.7004397 -4.700440 -4.700440 -4.700440  1.957772 -4.700440

```

```

> icm = toICM(pfm)
> icm

An object of class ICMatrix
ID: MA0004.1
Name: Arnt
Matrix Class: Zipper-Type
strand: +
Pseudocounts: 0.8
Schneider correction: FALSE
Tags:
$family
[1] "Helix-Loop-Helix"

$species
[1] "10090"

$tax_group
[1] "vertebrates"

$medline
[1] "7592839"

$type
[1] "SELEX"

$ACC
[1] "P53762"

$pazar_tf_id
[1] "TF0000003"

$TFBSshape_ID
[1] "11"

$TFencyclopedia_ID
[1] "580"

Background:
  A   C   G   T
0.25 0.25 0.25 0.25
Matrix:
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
A 0.22700966 1.40964891 0.01697796 0.01697796 0.01697796 0.01697796
C 0.87560869 0.01468384 1.71477409 0.01697796 0.01697796 0.01697796
G 0.01080998 0.08810306 0.01697796 1.71477409 0.01697796 1.71477409
T 0.01080998 0.01468384 0.01697796 0.01697796 1.71477409 0.01697796

> ## get the reverse complement matrix with all the same information except the strand.
> reverseComplement(pwm)

An object of class PWMMatrix
ID: MA0004.1
Name: Arnt
Matrix Class: Zipper-Type
strand: -

```

```

Pseudocounts: 0.8
Tags:
$family
[1] "Helix-Loop-Helix"

$species
[1] "10090"

$tax_group
[1] "vertebrates"

$medline
[1] "7592839"

$type
[1] "SELEX"

$ACC
[1] "P53762"

$pazar_tf_id
[1] "TF0000003"

$TFBSshape_ID
[1] "11"

$TFencyclopedia_ID
[1] "580"

Background:
  A   C   G   T
0.25 0.25 0.25 0.25
Matrix:
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
A -4.700440  1.957772 -4.700440 -4.700440 -4.700440 -4.7004397
C  1.957772 -4.700440  1.957772 -4.700440 -2.115477 -4.7004397
G -4.700440 -4.700440 -4.700440  1.957772 -4.700440  1.6394103
T -4.700440 -4.700440 -4.700440 -4.700440  1.884523 -0.3081223

```

2.2 XMatrixList and its subclasses

XMatrixList is used to store a set of *XMatrix* objects. Basically it is a *SimpleList* for easy manipulation the whole set of *XMatrix*. The concrete objects can be *PFMatrix*, *PWMMatrix* and *ICMatrix*.

```

> pfm2 = pfm
> pfmList = PFMatrixList(pfm1=pfm, pfm2=pfm2, use.names=TRUE)
> pfmList

PFMatrixList of length 2
names(2): pfm1 pfm2

> names(pfmList)

[1] "pfm1" "pfm2"

```

2.3 SiteSet, SiteSetList, SitePairSet and SitePairSetList

The *SiteSet* class is a container for storing a set of putative transcription factor binding sites on a nucleotide sequence (start, end, strand, score, pattern as a *PWMMatrix*, etc.) from scanning a nucleotide sequence with the corresponding *PWMMatrix*.

Similarly, *SiteSetList* stores a set of *SiteSet* objects. To store the results from scanning a pairwise alignment, *SitePairSet* is created.

2.4 MotifSet

This *MotifSet* class is used to store the generated motifs from motifs discovery software, such as *MEME*.

3 Database interfaces for JASPAR2014 database

This section will demonstrate how to operate on the JASPAR 2014 database. JASPAR is a collection of transcription factor DNA-binding preferences, modeled as matrices. These can be converted into Position Weight Matrices (PWMs or PSSMs), used for scanning genomic sequences. JASPAR is the only database with this scope where the data can be used with no restrictions (open-source).

3.1 Search JASPAR2014 database

This search function fetches matrix data for all matrices in the database matching criteria defined by the named arguments and returns a *PFMatrixList* object. For more search criterias, please see the help page for (*getMatrixSet*).

```
> library(JASPAR2014)
> opts = list()
> opts[["species"]] = 9606
> opts[["name"]] = "RUNX1"
> #opts[["class"]] = "Ig-fold"
> opts[["type"]] = "SELEX"
> opts[["all_versions"]] = TRUE
> PFMatrixList = getMatrixSet(JASPAR2014, opts)
> PFMatrixList

PFMatrixList of length 1
names(1): MA0002.1

> opts2 = list()
> opts2[["type"]] = "SELEX"
> PFMatrixList2 = getMatrixSet(JASPAR2014, opts2)
> PFMatrixList2

PFMatrixList of length 111
names(111): MA0004.1 MA0006.1 MA0008.1 MA0009.1 ... MA0587.1 MA0588.1 MA0589.1 MA0590.1
```

3.2 Store, delete and initialize JASPAR2014 database

We also provide some functions to initialize an empty JASPAR2014 style database, store new *PFMatrix* or *PFMatrixList* into it, or delete some records based on ID.

```
> db = "myMatrixDb.sqlite"
> initializeJASPARDB(db)
> storeMatrix(db, pfm)
> deleteMatrixHavingID(db, "MA0003")
```

4 PFM, PWM and ICM methods

This section will give an introduction of matrix conversion from PFM.

4.1 PFM to PWM

The method `toPWM` can convert PFM to PWM ([Wasserman and Sandelin, 2004](#)). Optional parameters include `type`, `pseudocounts`, `bg`. The implementation in this package is a bit different from *Biostrings*.

First of all, `toPWM` allows the input matrix to have different column sums, which means the count matrix can have an unequal number of sequences contributing to each column. This scenario is rare, but exists in JASPAR SELEX data.

Second, we can specify customized pseudocounts. `pseudocounts` is necessary for correcting the small number of counts or eliminating the zero values before log transformation. In TFBS perl module, the square root of the number of sequences contributing to each column. However, it has been shown to too harsh ([Nishida et al., 2009](#)). Hence, a default value of 0.8 is used. Of course, it can be changed to other customized value or even different values for each column.

```
> pwm = toPWM(pfm, pseudocounts=0.8)
> pwm
```

An object of class PWMMatrix

```
ID: MA0004.1
Name: Arnt
Matrix Class: Zipper-Type
strand: +
Pseudocounts: 0.8
Tags:
$family
[1] "Helix-Loop-Helix"
```

```
$species
[1] "10090"
```

```
$tax_group
[1] "vertebrates"
```

```
$medline
[1] "7592839"
```

```
$type
[1] "SELEX"
```

```
$ACC
[1] "P53762"
```

```
$pazar_tf_id
[1] "TF0000003"
```

```
$TFBSshape_ID
[1] "11"
```

```
$TFencyclopedia_ID
[1] "580"
```

```
Background:
```

```
  A   C   G   T
0.25 0.25 0.25 0.25
```

```
Matrix:
```

```
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
A -0.3081223  1.884523 -4.700440 -4.700440 -4.700440 -4.700440
C  1.6394103 -4.700440  1.957772 -4.700440 -4.700440 -4.700440
G -4.7004397 -2.115477 -4.700440  1.957772 -4.700440  1.957772
T -4.7004397 -4.700440 -4.700440 -4.700440  1.957772 -4.700440
```

4.2 PFM to ICM

The method `toICM` can convert PFM to ICM ([Schneider et al., 1986](#)). Besides the similar pseudocounts, `bg`, you can also choose to do the `schneider` correction.

```
> icm = toICM(pfm, pseudocounts=0.8, schneider=TRUE)
> icm
```

```
An object of class ICMatrix
```

```
ID: MA0004.1
```

```
Name: Arnt
```

```
Matrix Class: Zipper-Type
```

```
strand: +
```

```
Pseudocounts: 0.8
```

```
Schneider correction: TRUE
```

```
Tags:
```

```
$family
```

```
[1] "Helix-Loop-Helix"
```

```
$species
```

```
[1] "10090"
```

```
$tax_group
```

```
[1] "vertebrates"
```

```
$medline
```

```
[1] "7592839"
```

```
$type
```

```
[1] "SELEX"
```

```
$ACC
```

```
[1] "P53762"
```

```
$pazar_tf_id
```

```
[1] "TF0000003"
```

```
$TFBSshape_ID
```



```
[1] "11"
```

```
$TFencyclopedia_ID
```

```
[1] "580"
```

```
Background:
```

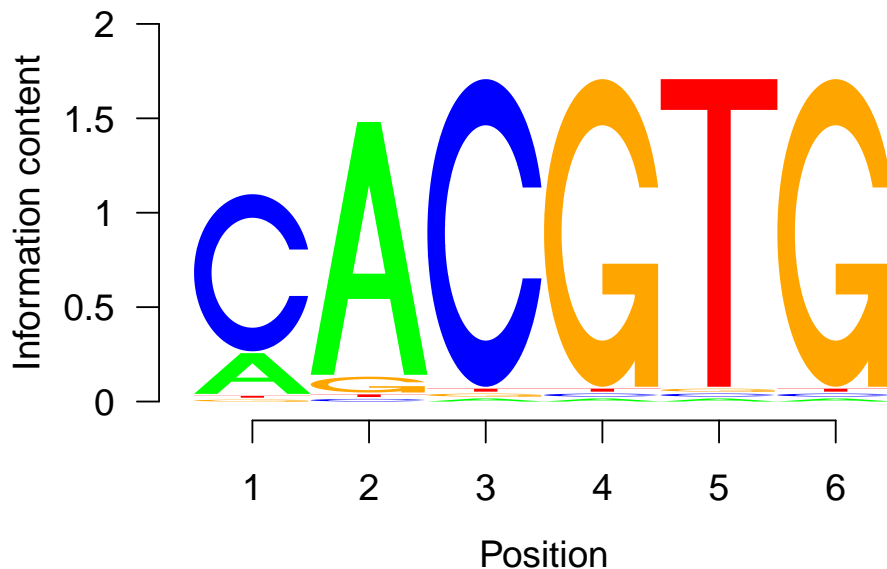
```
  A   C   G   T
0.25 0.25 0.25 0.25
```

```
Matrix:
```

```
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
A 0.203985791 1.30439694 0.01588159 0.01588159 0.01588159 0.01588159
C 0.786802337 0.01358747 1.60404024 0.01588159 0.01588159 0.01588159
G 0.009713609 0.08152481 0.01588159 1.60404024 0.01588159 1.60404024
T 0.009713609 0.01358747 0.01588159 0.01588159 1.60404024 0.01588159
```

To plot the sequence logo, we use the package *seqlogo*.

```
> seqLogo(icm)
```



4.3 Align PFM to a custom matrix or IUPAC string

In some cases, it is beneficial to assess similarity to input data (as with using BLAST for sequence data comparison when using Genbank).

```
> ## one to one comparison
> data(MA0003.2)
> data(MA0004.1)
> pfmSubject = MA0003.2
> pfmQuery = MA0004.1
> PFMSimilarity(pfmSubject, pfmQuery)
```

```

      score relScore
8.103627 27.012091
> ## one to several comparsion
> PFMSimilarity(pfmList, pfmQuery)
$pfm1
  score relScore
    12     100

$pfm2
  score relScore
    12     100

> ## align IUPAC string
> IUPACString = "ACGTMRWSYKVHDBN"
> PFMSimilarity(pfmList, IUPACString)
$pfm1
  score relScore
8.815001 29.383335

$pfm2
  score relScore
8.815001 29.383335

```

4.4 PFM permutation

In this section, we will demonstrate the matrix permutation. This method simply shuffles the columns in matrices. This can either be done by just shuffling columns within each selected matrix, or by shuffling columns among all selected matrices.

```

> #library(JASPAR2014)
> #pfmSubject = getMatrixByID(JASPAR2014, ID="MA0043")
> #pfmQuery = getMatrixByID(JASPAR2014, ID="MA0048")
> #opts = list()
> #opts[["class"]] = "Ig-fold"
> #pfmList = getMatrixSet(JASPAR2014, opts)
> permuteMatrix(pfmQuery)

```

```

An object of class PFMatrix
ID: MA0004.1
Name: Arnt
Matrix Class: Zipper-Type
strand: +
Tags:
$comment
[1] "-"

$family
[1] "Helix-Loop-Helix"

$medline
[1] "7592839"

$pazar_tf_id

```

```

[1] "TF0000003"

$tax_group
[1] "vertebrates"

$tfbs_shape_id
[1] "11"

$tfe_id
[1] "580"

$type
[1] "SELEX"

$collection
[1] "CORE"

$species
[1] "10090"

$acc
[1] "P53762"

Background:
  A   C   G   T
0.25 0.25 0.25 0.25
Matrix:
  [,1] [,2] [,3] [,4] [,5] [,6]
A    0    0   19    0    0    4
C    0    0    0    0   20   16
G    0   20    1   20    0    0
T   20    0    0    0    0    0

> permuteMatrix(pfmList, type="intra")
PFMatrixList of length 2
names(2): pfm1 pfm2

> permuteMatrix(pfmList, type="inter")
PFMatrixList of length 2
names(2): pfm1 pfm2

```

4.5 PWM similarity

To measure the similarity of two PWM matrix in three measurements: "normalised Euclidean distance", "Pearson correlation" and "Kullback Leibler divergence" (Linhart et al., 2008). Given two PWMs, P^1 and P^2 , where l is the length. $P_{i,b}$ is the values in column i with base b . The normalised Euclidean distance is computed in

$$D(a,b) = \frac{1}{\sqrt{2}l} \cdot \sum_{i=1}^l \sqrt{\sum_{b \in \{A,C,G,T\}} (P_{i,b}^1 - P_{i,b}^2)^2} \quad (1)$$

This distance is between 0 (perfect identity) and 1 (complete dis-similarity).

The pearson correlation coefficient is computed in

$$r(P^1, P^2) = \frac{1}{l} \cdot \sum_{i=1}^l \frac{\sum_{b \in \{A,C,G,T\}} (P_{i,b}^1 - 0.25)(P_{i,b}^2 - 0.25)}{\sqrt{\sum_{b \in \{A,C,G,T\}} (P_{i,b}^1 - 0.25)^2 \cdot \sum_{b \in \{A,C,G,T\}} (P_{i,b}^2 - 0.25)^2}}. \quad (2)$$

The Kullback-Leibler divergence is computed in

$$KL(P^1, P^2) = \frac{1}{2l} \cdot \sum_{i=1}^l \sum_{b \in \{A,C,G,T\}} (P_{i,b}^1 \log \frac{P_{i,b}^1}{P_{i,b}^2} + P_{i,b}^2 \log \frac{P_{i,b}^2}{P_{i,b}^1}). \quad (3)$$

```
> data(MA0003.2)
> data(MA0004.1)
> pwm1 = toPWM(MA0003.2, type="prob")
> pwm2 = toPWM(MA0004.1, type="prob")
> PWMSimilarity(pwm1, pwm2, method="Euclidean")
[1] 0.5134956
> PWMSimilarity(pwm1, pwm2, method="Pearson")
[1] 0.2828507
> PWMSimilarity(pwm1, pwm2, method="KL")
[1] 2.385866
```

5 Scan sequence and alignments with PWM pattern

5.1 searchSeq

searchSeq scans a nucleotide sequence with the pattern represented by the PWM. The strand argument controls which strand of the sequence will be searched. When it is "*", both strands will be scanned.

```
> library(Biostrings)
> data(MA0003.2)
> data(MA0004.1)
> pwmList = PWMMatrixList(MA0003.2=toPWM(MA0003.2), MA0004.1=toPWM(MA0004.1),
+ use.names=TRUE)
> subject = DNASTring("GAATTCTCTCTTGTGTAGTCTCTTGACAAAATG")
> siteset = searchSeq(pwm, subject, seqname="seq1", min.score="60%", strand="*")
> sitesetList = searchSeq(pwmList, subject, seqname="seq1",
+ min.score="60%", strand="*")
> ## generate gff style output
> head(writeGFF3(siteset))

seqname source feature start end score strand frame
1 seq1 TFBS TFBS 8 13 -1.888154 + .
2 seq1 TFBS TFBS 21 26 -1.888154 + .
3 seq1 TFBS TFBS 29 34 -3.908935 + .
4 seq1 TFBS TFBS 8 13 -1.961403 - .
5 seq1 TFBS TFBS 10 15 -3.908935 - .
6 seq1 TFBS TFBS 21 26 -1.961403 - .
attributes
1 TF=Arnt;class=Zipper-Type;sequence=CTCTTG
2 TF=Arnt;class=Zipper-Type;sequence=CTCTTG
```

```

3 TF=Arnt;class=Zipper-Type;sequence=AAAATG
4 TF=Arnt;class=Zipper-Type;sequence=CAAGAG
5 TF=Arnt;class=Zipper-Type;sequence=AACAAG
6 TF=Arnt;class=Zipper-Type;sequence=CAAGAG

> head(writeGFF3(sitesetList))

      seqname source feature start end      score strand frame
MA0003.2   seq1  TFBS    TFBS    18  32 -16.437682      -      .
MA0004.1.1 seq1  TFBS    TFBS     8  13  -1.888154      +      .
MA0004.1.2 seq1  TFBS    TFBS    21  26  -1.888154      +      .
MA0004.1.3 seq1  TFBS    TFBS    29  34  -3.908935      +      .
MA0004.1.4 seq1  TFBS    TFBS     8  13  -1.961403      -      .
MA0004.1.5 seq1  TFBS    TFBS    10  15  -3.908935      -      .

                                attributes
MA0003.2 TF=TFAP2A;class=Zipper-Type;sequence=TTTTGTCAAGAGACT
MA0004.1.1 TF=Arnt;class=Zipper-Type;sequence=CTCTTG
MA0004.1.2 TF=Arnt;class=Zipper-Type;sequence=CTCTTG
MA0004.1.3 TF=Arnt;class=Zipper-Type;sequence=AAAATG
MA0004.1.4 TF=Arnt;class=Zipper-Type;sequence=CAAGAG
MA0004.1.5 TF=Arnt;class=Zipper-Type;sequence=AACAAG

> head(writeGFF2(siteset))

      seqname source feature start end      score strand frame
1      seq1  TFBS    TFBS     8  13 -1.888154      +      .
2      seq1  TFBS    TFBS    21  26 -1.888154      +      .
3      seq1  TFBS    TFBS    29  34 -3.908935      +      .
4      seq1  TFBS    TFBS     8  13 -1.961403      -      .
5      seq1  TFBS    TFBS    10  15 -3.908935      -      .
6      seq1  TFBS    TFBS    21  26 -1.961403      -      .

                                attributes
1 TF "Arnt"; class "Zipper-Type"; sequence "CTCTTG"
2 TF "Arnt"; class "Zipper-Type"; sequence "CTCTTG"
3 TF "Arnt"; class "Zipper-Type"; sequence "AAAATG"
4 TF "Arnt"; class "Zipper-Type"; sequence "CAAGAG"
5 TF "Arnt"; class "Zipper-Type"; sequence "AACAAG"
6 TF "Arnt"; class "Zipper-Type"; sequence "CAAGAG"

> ## get the relative score
> relScore(siteset)

[1] 0.6652185 0.6652185 0.6141340 0.6633668 0.6141340 0.6633668

> relScore(sitesetList)

$MA0003.2
[1] 0.6196884

$MA0004.1
[1] 0.6652185 0.6652185 0.6141340 0.6633668 0.6141340 0.6633668

```

5.2 searchAln

`searchAln` scans a pairwise alignment with the pattern represented by the PWM. It reports only those hits that are present in equivalent positions of both sequences and exceed a specified threshold score in both, AND are found in regions of the alignment above the specified.

```

> library(Biostrings)
> data(MA0003.2)
> pwm = toPWM(MA0003.2)
> aln1 = DNASTring("ACTTCACCAGCTCCCTGGCGGTAAGTTGATC---AAAGG---AAACGCAAAGTTTTCAAG")
> aln2 = DNASTring("GTTTCACTACTTCCTTTTCGGGTAAGTAAATATATAAAATATATAAAAAATATAATTTTCATC")
> sitePairSet = searchAln(pwm, aln1, aln2, seqname1="seq1", seqname2="seq2",
+                         min.score="50%", cutoff=0.5,
+                         strand="*", type="any")
> ## generate gff style output
> head(writeGFF3(sitePairSet))

```

	seqname	source	feature	start	end	score	strand	frame
1	seq1	TFBS	TFBS	6	20	-9.515444	+	.
2	seq1	TFBS	TFBS	7	21	-13.348617	+	.
3	seq1	TFBS	TFBS	8	22	-13.182322	+	.
4	seq1	TFBS	TFBS	9	23	-3.729917	+	.
5	seq1	TFBS	TFBS	10	24	-7.677850	+	.
6	seq1	TFBS	TFBS	14	28	-20.774619	+	.

attributes

```

1 TF=TFAP2A;class=Zipper-Type;sequence=ACCAGCTCCCTGGCG
2 TF=TFAP2A;class=Zipper-Type;sequence=CCAGCTCCCTGGCGG
3 TF=TFAP2A;class=Zipper-Type;sequence=CAGCTCCCTGGCGGT
4 TF=TFAP2A;class=Zipper-Type;sequence=AGCTCCCTGGCGGTA
5 TF=TFAP2A;class=Zipper-Type;sequence=GCTCCCTGGCGGTAA
6 TF=TFAP2A;class=Zipper-Type;sequence=CCTGGCGGTAAGTTG

```

```

> head(writeGFF2(sitePairSet))

```

	seqname	source	feature	start	end	score	strand	frame
1	seq1	TFBS	TFBS	6	20	-9.515444	+	.
2	seq1	TFBS	TFBS	7	21	-13.348617	+	.
3	seq1	TFBS	TFBS	8	22	-13.182322	+	.
4	seq1	TFBS	TFBS	9	23	-3.729917	+	.
5	seq1	TFBS	TFBS	10	24	-7.677850	+	.
6	seq1	TFBS	TFBS	14	28	-20.774619	+	.

attributes

```

1 TF "TFAP2A"; class "Zipper-Type"; sequence "ACCAGCTCCCTGGCG"
2 TF "TFAP2A"; class "Zipper-Type"; sequence "CCAGCTCCCTGGCGG"
3 TF "TFAP2A"; class "Zipper-Type"; sequence "CAGCTCCCTGGCGGT"
4 TF "TFAP2A"; class "Zipper-Type"; sequence "AGCTCCCTGGCGGTA"
5 TF "TFAP2A"; class "Zipper-Type"; sequence "GCTCCCTGGCGGTAA"
6 TF "TFAP2A"; class "Zipper-Type"; sequence "CCTGGCGGTAAGTTG"

```

```

> ## search the Axt alignment

```

```

> library(CNER)
> axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="CNER"),
+                                 "hg19.danRer7.net.axt")
> axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)

```

```

The number of axt files 1

```

```

The number of axt alignments is 133

```

```

> sitePairSet <- searchAln(pwm, axtHg19DanRer7, min.score="80%",
+                          windowSize=51L, cutoff=0.7, strand="*",
+                          type="any", conservation=NULL, mc.cores=2)
> GRangesTFBS <- toGRangesList(sitePairSet, axtHg19DanRer7)
> GRangesTFBS$targetTFBS

```

GRanges object with 33 ranges and 5 metadata columns:

	seqnames <Rle>	ranges <IRanges>	strand <Rle>	matrix.ID <character>	matrix.strand <character>	abs.score <numeric>
[1]	chr11	[31128026, 31128040]	+	MA0003.2	+	2.089093
[2]	chr11	[31128028, 31128042]	+	MA0003.2	-	1.404957
[3]	chr11	[31437161, 31437175]	+	MA0003.2	+	7.785995
[4]	chr11	[31437163, 31437177]	+	MA0003.2	-	7.796633
[5]	chr11	[31499628, 31499642]	+	MA0003.2	+	2.239032
...
[29]	chr11	[32198154, 32198168]	+	MA0003.2	+	2.511814
[30]	chr11	[32221916, 32221930]	+	MA0003.2	+	4.672261
[31]	chr11	[32221918, 32221932]	+	MA0003.2	-	10.609562
[32]	chr11	[32456353, 32456367]	+	MA0003.2	+	4.204543
[33]	chr11	[32456355, 32456369]	+	MA0003.2	-	1.176965

	rel.score <numeric>	sitesSeq <DNASTringSet>
[1]	0.8130353	GGCCCCCGTGGCGT
[2]	0.8058956	CCCCCGTGGCGTTT
[3]	0.8724886	ACTGGCTAGGGACA
[4]	0.8725997	TGGCCTAGGGACAGG
[5]	0.8146001	TCTGACCCAGAGGCA
...
[29]	0.8174469	ACCGTCTTTAGGGGA
[30]	0.8399935	TCTACCCTCGAGCAC
[31]	0.9019556	TACCCTCGAGCACTG
[32]	0.8351123	AAGGGCCCGTAGCGA
[33]	0.8035163	GGGCCCGTAGCGACA

seqinfo: 3 sequences from an unspecified genome; no seqlengths

> GRangesTFBS\$queryTFBS

GRanges object with 33 ranges and 5 metadata columns:

	seqnames <Rle>	ranges <IRanges>	strand <Rle>	matrix.ID <character>	matrix.strand <character>	abs.score <numeric>
[1]	chr25	[15057430, 15057444]	+	MA0003.2	+	2.656516
[2]	chr25	[15057432, 15057446]	+	MA0003.2	-	5.005684
[3]	chr25	[15149515, 15149529]	+	MA0003.2	+	5.686105
[4]	chr25	[15149517, 15149531]	+	MA0003.2	-	6.434853
[5]	chr25	[15163682, 15163696]	+	MA0003.2	+	1.492573
...
[29]	chr25	[15359169, 15359183]	+	MA0003.2	+	3.134736
[30]	chr25	[15472022, 15472036]	+	MA0003.2	+	1.340569
[31]	chr25	[15472024, 15472038]	+	MA0003.2	-	5.674186
[32]	chr25	[15570786, 15570800]	+	MA0003.2	+	1.604621
[33]	chr25	[15570789, 15570803]	+	MA0003.2	-	5.261771

	rel.score <numeric>	sitesSeq <DNASTringSet>
[1]	0.8189570	GGCCCCCTGCGGCTC
[2]	0.8434731	CCCCCTGCGGCTCTC
[3]	0.8505740	TCTCGCTAAGGAAA
[4]	0.8583880	TCGCCTAAGGAAAAA
[5]	0.8068100	TGCCACCCTTGGGCA
...
[29]	0.8239477	ACCATCTTTAGGGGA

```
[30] 0.8052237 TCTCCCCTCCAGCTC
[31] 0.8504496 TCCCCTCCAGCTCTG
[32] 0.8079793 CCGTACCTGCAGGCC
[33] 0.8461456 TACCTGCAGGCCCT
-----
```

```
seqinfo: 3 sequences from an unspecified genome; no seqlengths
```

5.3 searchPairBSgenome

`searchPairBSgenome` is designed to do the genome-wise phylogenetic footprinting. Given two *BSgenome*, a chain file for liftover from one genome to another, `searchPairBSgenome` identifies the putative transcription factor binding sites which are conserved in both genomes.

```
> library(rtracklayer)
> library(JASPAR2014)
> library(BSgenome.Hsapiens.UCSC.hg19)
> library(BSgenome.Mmusculus.UCSC.mm10)
> pfm = getMatrixByID(JASPAR2014, ID="MA0004.1")
> pwm=toPWM(pfm)
> chain = import.chain("Downloads/hg19ToMm10.over.chain")
> sitePairSet = searchPairBSgenome(pwm, BSgenome.Hsapiens.UCSC.hg19,
+                               BSgenome.Mmusculus.UCSC.mm10,
+                               chr1="chr1", chr2="chr1",
+                               min.score="90%", strand="+", chain=chain)
```

6 Use external pattern generators

In this section, we will introduce wrapper functions for external motif discovery programs. So far, MEME is supported.

6.1 MEME

`runMEME` takes a *DNAStringSet* or a set of *characters* as input, and returns a *MotifSet* object.

```
> motifSet = runMEME(file.path(system.file("extdata",
+                                       package="TFBSTools"), "crp0.s"),
+                   binary="meme",
+                   arguments=list("-nmotifs=3")
+                   )
> sitesSeq(motifSet, type="all")
> sitesSeq(motifSet, type="none")
> consensusMatrix(motifSet)
```

7 Conclusion

The following is the session info that generated this vignette:

```
> sessionInfo()

R version 3.1.1 Patched (2014-09-25 r66681)
Platform: x86_64-unknown-linux-gnu (64-bit)
```


locale:

```
[1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C              LC_TIME=en_US.UTF-8
[4] LC_COLLATE=C              LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8     LC_NAME=C                 LC_ADDRESS=C
[10] LC_TELEPHONE=C           LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

attached base packages:

```
[1] stats4    parallel  stats     graphics  grDevices  utils      datasets  methods
[9] base
```

other attached packages:

```
[1] CNEr_1.2.0          JASPAR2014_1.1.0      Biostrings_2.34.0    XVector_0.6.0
[5] IRanges_2.0.0      S4Vectors_0.4.0      BiocGenerics_0.12.0 TFBSTools_1.4.0
```

loaded via a namespace (and not attached):

```
[1] BBmisc_1.7          BSgenome_1.34.0      BatchJobs_1.4
[4] BiocParallel_1.0.0 BiocStyle_1.4.0      DBI_0.3.1
[7] DirichletMultinomial_1.8.0 GenomeInfoDb_1.2.0   GenomicAlignments_1.2.0
[10] GenomicRanges_1.18.0 RCurl_1.95-4.3      RSQLite_0.11.4
[13] Rcpp_0.11.3         Rsamtools_1.18.0     TFMpvalue_0.0.5
[16] XML_3.98-1.1        base64enc_0.1-2      bitops_1.0-6
[19] brew_1.0-6          caTools_1.17.1      checkmate_1.4
[22] codetools_0.2-9     digest_0.6.4         fail_1.2
[25] foreach_1.4.2       grid_3.1.1           gtools_3.4.1
[28] iterators_1.0.7     rtracklayer_1.26.0  sendmailR_1.2-1
[31] seqLogo_1.32.0     stringr_0.6.2        tools_3.1.1
[34] zlibbioc_1.12.0
```

References

- T L Bailey and C Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol*, 2:28–36, 1994. ISSN 1553-0833. PMID: 7584402.
- Boris Lenhard and Wyeth W Wasserman. TFBS: computational framework for transcription factor binding site analysis. *Bioinformatics*, 18(8):1135–1136, August 2002. ISSN 1367-4803. PMID: 12176838.
- Chaim Linhart, Yonit Halperin, and Ron Shamir. Transcription factor and microRNA motif discovery: the amadeus platform and a compendium of metazoan target sets. *Genome Res.*, 18(7):1180–1189, July 2008. ISSN 1088-9051. doi: 10.1101/gr.076117.108. PMID: 18411406.
- Keishin Nishida, Martin C Frith, and Kenta Nakai. Pseudocounts for transcription factor binding sites. *Nucleic Acids Res.*, 37(3):939–944, February 2009. ISSN 1362-4962. doi: 10.1093/nar/gkn1019. PMID: 19106141.
- T D Schneider, G D Stormo, L Gold, and A Ehrenfeucht. Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188(3):415–431, April 1986. ISSN 0022-2836. PMID: 3525846.
- Wyeth W Wasserman and Albin Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature Publishing Group*, 5(4):276–287, April 2004.