# atSNP: affinity tests for regulatory SNP detection

Chandler Zuo, Sunyoung Shin and Sündüz Keleş
Department of Statistics and of Biostatistics and Medical Informatics, University of Wisconsin Madison

## Contents



## 1 Introduction

This document provides an introduction to the affinity test for large sets of SNP-motif interactions using the *atSNP* package(**a**ffinity **t**est for regulatory **SNP** detection) [5]. *atSNP* implements in-silico methods for identifying SNPs that potentially may affect binding affinity of transcription factors. Given a set of SNPs and a library of motif position weight matrices (PWMs), *atSNP* provides three main functions for analyzing SNP effects:

1. Computing the binding affinity score for each allele and each PWM.
2. Computing the p-values for allele-specific binding affinity scores.
3. Computing the p-values for affinity score changes between the two alleles for each SNP.

*atSNP* implements the importance sampling algorithm in [1] to compute the p-values. Compared to other bioinformatics tools, such as FIMO [2] and is-rSNP [4] that provide similar functionalities, *atSNP* avoids computing the p-values analytically. In one of our research projects, we have used atSNP to evaluate interactions between 26K SNPs and 2K motifs within 5 hours. We found no other existing tool can finish the analysis of such a scale.

## 2 Installation

We are working to make the package available through bioconductor. The developing version can be installed from the Github repository:

```
library(devtools)
install_github("chandlerzuo/atSNP")
```

*atSNP* depends on the following *R* packages:

- *data.table* is used for formatting results that are easy for users to query.
- *motifStack* is relied upon to draw sequence logo plots.
- *doParallel* and *foreach* are used for parallel computation.
- *Rcpp* interfaces the C++ codes that implements the importance sampling algorithm.

In addition, users need to install the annotation package *BSgenome* from `www.bioconductor.org/packages/3.0/data/annotation/` that corresponds to the species type and genome version. Our example SNP data set in the subsequent sections corresponds to the hg19 version of human genome. To repeat the sample codes in this vignette, the *BSgenome.Hsapiens.UCSC.hg19* package is required. If users wish to annotate the SNP location and allele information given their rs ids, they also need install the corresponding *SNPlocs* package. The sample codes in this vignettes require the package *SNPlocs.Hsapiens.dbSNP.20120608*. To install these packages from the *Bioconductor* repository,

```
source("http://bioconductor.org/biocLite.R")
biocLite("BSgenome.Hsapiens.UCSC.hg19")
biocLite("SNPlocs.Hsapiens.dbSNP.20120608")
```

Notice that the annotation packages are usually large and this installation step may take a substantial amount of time.

# 3 Example

## 3.1 Load the motif library

*atSNP* includes two motif libraries in the package: the ENCODE derived motif library, and the JASPAR database motif library. In addition, *atSNP* can load user defined motif libraries in a variety of formats.

### 3.1.1 ENCODE derived motif library

*atSNP* provides a default motif library downloaded from `compbio.mit.edu/encode-motifs/motifs.txt`. This library contains 2065 known and discovered motifs from ENCODE TF ChIP-seq data sets. The following commands allows to load this motif library:

```
library(atSNP)
```

```
## Loading required package: Rcpp
## Loading required package: data.table
##
## Attaching package: 'data.table'
##
## The following object is masked from 'package:GenomicRanges':
##
##     last
##
## Loading required package: doParallel
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: motifStack
## Loading required package: grImport
## Loading required package: grid
## Loading required package: XML
## Loading required package: MotIV
##
## Attaching package: 'MotIV'
##
```

```
## The following object is masked from 'package:stats':
##
##    filter
##
## Loading required package:  ade4
##
## Attaching package:  'ade4'
##
## The following object is masked from 'package:BSgenome':
##
##    score
##
## The following object is masked from 'package:Biostrings':
##
##    score
##
## The following object is masked from 'package:GenomicRanges':
##
##    score
##
## The following object is masked from 'package:IRanges':
##
##    score
```

```
data(encode_library)
length(encode_motif)

## [1] 2065

encode_motif[1]

## $SIX5_disc1
##              [,1]       [,2]     [,3]        [,4]
##  [1,] 8.51100e-03 4.2550e-03 0.987234 1.00000e-10
##  [2,] 9.02127e-01 1.2766e-02 0.038298 4.68090e-02
##  [3,] 4.55319e-01 7.2340e-02 0.344681 1.27660e-01
##  [4,] 2.51064e-01 8.5106e-02 0.085106 5.78724e-01
##  [5,] 1.00000e-10 4.6809e-02 0.012766 9.40425e-01
##  [6,] 1.00000e-10 1.0000e-10 1.000000 1.00000e-10
##  [7,] 3.82980e-02 2.1277e-02 0.029787 9.10638e-01
##  [8,] 9.44681e-01 4.2550e-03 0.051064 1.00000e-10
##  [9,] 1.00000e-10 1.0000e-10 1.000000 1.00000e-10
## [10,] 1.00000e-10 1.0000e-10 0.012766 9.87234e-01
```

Here, the motif library is represented by `encode_motif`, which is a list of position weight matrices. The codes below shows the content of one matrix as well as its IUPAC letters:

```
encode_motif[[1]]

##              [,1]       [,2]     [,3]        [,4]
##  [1,] 8.51100e-03 4.2550e-03 0.987234 1.00000e-10
##  [2,] 9.02127e-01 1.2766e-02 0.038298 4.68090e-02
##  [3,] 4.55319e-01 7.2340e-02 0.344681 1.27660e-01
##  [4,] 2.51064e-01 8.5106e-02 0.085106 5.78724e-01
##  [5,] 1.00000e-10 4.6809e-02 0.012766 9.40425e-01
##  [6,] 1.00000e-10 1.0000e-10 1.000000 1.00000e-10
##  [7,] 3.82980e-02 2.1277e-02 0.029787 9.10638e-01
##  [8,] 9.44681e-01 4.2550e-03 0.051064 1.00000e-10
```

```
##  [9,] 1.00000e-10 1.0000e-10 1.000000 1.00000e-10
## [10,] 1.00000e-10 1.0000e-10 0.012766 9.87234e-01

GetIUPACSequence(encode_motif[[1]])

## [1] "GARWTGTAGT"
```

The data object `encode_library` also contains a character vector `encode_motifinfo` that contains detailed information for each motif.

```
length(encode_motifinfo)

## [1] 2065

head(encode_motifinfo)

##                                                SIX5_disc1
##   "SIX5_GM12878_encode-Myers_seq_hsa_r1:MEME#1#Intergenic"
##                                                 MYC_disc1
##   "USF2_K562_encode-Snyder_seq_hsa_r1:MDscan#1#Intergenic"
##                                                 SRF_disc1
##  "SRF_H1-hESC_encode-Myers_seq_hsa_r1:MDscan#2#Intergenic"
##                                                 AP1_disc1
##     "JUND_K562_encode-Snyder_seq_hsa_r1:MEME#1#Intergenic"
##                                                SIX5_disc2
## "SIX5_H1-hESC_encode-Myers_seq_hsa_r1:MDscan#1#Intergenic"
##                                                 NFY_disc1
##     "NFYA_K562_encode-Snyder_seq_hsa_r1:MEME#2#Intergenic"
```

Here, the entry names of this vector are the same as the names of the motif library. `encode_motifinfo` allows easy looking up the motif information for a specific PWM. For example, to look up the motif information for the first PWM in `encode_motifinfo`:

```
encode_motifinfo[names(encode_motif[1])]

##                                                SIX5_disc1
## "SIX5_GM12878_encode-Myers_seq_hsa_r1:MEME#1#Intergenic"
```

### 3.1.2 JASPAR database motif library

Our package also includes the JASPAR library downloaded from `http://jaspar.genereg.net/html/DOWNLOAD/JASPAR_CORE/pfm/nonredundant/pfm_all.txt`. The data object `jaspar_library` contains a list of 593 PWMs `jaspar_motif` and a character vector `jaspar_motifinfo`.

```
data(jaspar_library)
jaspar_motif[[1]]

##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.20833333 0.70833333 0.04166667 0.04166667
## [2,] 0.83333333 0.04166667 0.08333333 0.04166667
## [3,] 0.04166667 0.87500000 0.04166667 0.04166667
## [4,] 0.04166667 0.04166667 0.87500000 0.04166667
## [5,] 0.04166667 0.04166667 0.04166667 0.87500000
## [6,] 0.04166667 0.04166667 0.87500000 0.04166667

jaspar_motifinfo[names(jaspar_motif[1])]

## MA0004.1
##   "Arnt"
```

### 3.1.3 User defined motif library

Users can also provide a list of PWMs as the motif library via the `LoadMotifLibrary` function. In this function, 'tag' specifies the string that marks the start of each block of PWM; 'skiprows' is the number of description lines before the PWM; 'skipcols' is the number of columns to be skipped in the PWM matrix; 'transpose' is TRUE if the PWM has 4 rows representing A, C, G, T or FALSE if otherwise; 'field' is the position of the motif name within the description line; 'sep' is a vector of separators in the PWM; 'pseudocount' is the number added to the raw matrices, recommended to be 1 if the matrices are in fact position frequency matrices. These arguments provide the flexibility of loading a number of varying formatted files. The PWMs are returned as a list object. This function flexibly adapts to a variety of different formats. Some examples using online accessible files from other research groups are shown below.

```
## Source: http://meme.nbcr.net/meme/doc/examples/sample-dna-motif.meme-io
pwms <- LoadMotifLibrary(
 "http://pages.stat.wisc.edu/~keles/atSNP-Data/sample-dna-motif.meme-io.txt")

## Source: http://compbio.mit.edu/encode-motifs/motifs.txt
pwms <- LoadMotifLibrary(
 "http://pages.stat.wisc.edu/~keles/atSNP-Data/motifs.txt",
 tag = ">", transpose = FALSE, field = 1,
 sep = c("\t", " ", ">"), skipcols = 1,
 skiprows = 1, pseudocount = 0)

## Source: http://johnsonlab.ucsf.edu/mochi_files/JASPAR_motifs_H_sapiens.txt
pwms <- LoadMotifLibrary(
 "http://pages.stat.wisc.edu/~keles/atSNP-Data/JASPAR_motifs_H_sapiens.txt",
 tag = "/NAME",skiprows = 1, skipcols = 0, transpose = FALSE,
 field = 2)

## Source: http://jaspar.genereg.net/html/DOWNLOAD/ARCHIVE/JASPAR2010/all_data/matrix_only/matrix.txt
pwms <- LoadMotifLibrary(
 "http://pages.stat.wisc.edu/~keles/atSNP-Data/matrix.txt",
 tag = ">", skiprows = 1, skipcols = 1, transpose = TRUE,
 field = 1, sep = c("\t", " ", "\\[", "\\]", ">"),
 pseudocount = 1)

## Source: http://jaspar.genereg.net/html/DOWNLOAD/JASPAR_CORE/pfm/nonredundant/pfm_vertebrates.txt
pwms <- LoadMotifLibrary(
 "http://pages.stat.wisc.edu/~keles/atSNP-Data/pfm_vertebrates.txt",
 tag = ">", skiprows = 1, skipcols = 0, transpose = TRUE, field = 1,
 sep = c(">", "\t", " "), pseudocount = 1)

## pwms <- LoadMotifLibrary(
##  "http://gibbs.biomed.ucf.edu/PreDREM/download/nonredundantmotif.transfac",
##  tag = "DE", skiprows = 1, skipcols = 1,
##  transpose = FALSE, field = 2, sep = "\t")
```

## 3.2 Load the SNP Data

*atSNP* can load the SNP data in three formats: a table including full SNP information, a list of dbSNP's rsids, and a pair of fasta files.

### 3.2.1 Load SNP data through a table

In this case, the table that provides the SNP information must include five columns:

- chr: the chromosome ID;
- snp: the genome coordinate of the SNP;
- snpid: the string for the SNP name;

- a1, a2: nucleotides for the two alleles at the SNP position.

This data set can be loaded using the LoadSNPData function. The 'genome.lib' argument specifies the annotation package name corresponding to the SNP data set, with the default as 'BSgenome.Hsapiens.UCSC.hg19'. Each side of the SNP is extended by a number of base pairs specified by the 'half.window.size' argument. LoadSNPData extracts the genome sequence within such windows around each SNP using the 'genome.lib' package. An example is the following:

The following codes generate a synthetic SNP data and loads it back in *R*:

```
data(example)
write.table(snp_tbl, file = "test_snp_file.txt",
            row.names = FALSE, quote = FALSE)
snp_info <- LoadSNPData("test_snp_file.txt", genome.lib = "BSgenome.Hsapiens.UCSC.hg19",
                        half.window.size = 30, default.par = TRUE, mutation = FALSE)

## Warning:  the following sequences are discarded because the reference nucleotide matches to
neither a1 nor a2:
## snpid chr snp a1 a2
## rs2495365 chr1 2498027 T C
## rs2281852 chr1 2490942 T G
## rs1886730 chr1 2488608 A G

ncol(snp_info$sequence) == nrow(snp_tbl)

## [1] FALSE

snp_info$rsid.rm

## [1] "rs2495365" "rs2281852" "rs1886730"
```

There are two important arguments in function LoadSNPData. First, the 'mutation' argument specifies whether the data set is related to SNP or general single nucleotide mutation. By default, 'mutation=FALSE'. In this case, LoadSNPData get the nucleotides on the reference genome based on the genome coordinates specified by 'chr' and 'snp' and match them to 'a1' and 'a2' alleles from the *BSgenome* package. 'a1' and 'a2' nucleotides are assigned to the refrence or the SNP allele based on which one matches to the reference nucleotide. If neither allele matches to the reference nucleotide, the corresponding row in the SNP information file is discarded. These discarded SNPs are captured by the 'rsid.rm' field in the output. Alternatively, if 'mutation=TRUE', no row is discarded. LoadSNPData takes the reference sequences around the SNP locations, replaces the reference nucleotides at the SNP locations by 'a1' nucleotides to construct the 'reference' sequences, and by 'a2' nucleotides to construct the 'SNP' sequences. Notice that in this case, in the subsequent analysis, whenever we refer to the "reference" or the "SNP" allele, it actually means the "a1" or the "a2" allele.

```
  mutation_info <- LoadSNPData("test_snp_file.txt", genome.lib = "BSgenome.Hsapiens.UCSC.hg19",
                               half.window.size = 30, default.par = TRUE, mutation = TRUE)
  ncol(mutation_info$sequence) == nrow(snp_tbl)

## [1] TRUE

  file.remove("test_snp_file.txt")

## [1] TRUE
```

Second, the 'default.par' argument specifies how to estimate the first order Markov model parameters. If 'default.par = FALSE', LoadSNPData simultaneously estimates the parameters for the first order Markov model in the reference genome using the nucleotides within the SNP windows. Otherwise, it loads a set of parameter values pre-fitted from sequences around all the SNPs in the NHGRI GWAS catalog ([3]). We recommend setting 'default.par = TRUE' when we have fewer than 1000 SNPs. LoadSNPData returns a list object with five fields:

- $sequence_matrix: a matrix with (2×'half.window.size' + 1), with each column corresponding to one SNP. The entries 1-4 represent the A, C, G, T nucleotides.
- $ref_base: a vector coding the reference allele nucleotides for all SNPs.
- $snp_base: a vector coding the SNP allele nucleotides for all SNPs.

- $prior: the stationary distribution parameters for the Markov model.
- $transition: the transition matrix for the first order Markov model.

Because LoadSNPData looks up the nucleotide at the SNP location from the *BSgenome* package,

### 3.2.2 Load SNP data through dbSNP's rsids

LoadSNPData also allows users to load a list of rsids for the SNPs. In this case, the function looks up the SNP location and the allele information using the annotation package specified by 'snp.lib'. The default value of 'snp.lib' is 'SNPlocs.Hsapiens.dbSNP.20120608'.

```
snp_info1 <- LoadSNPData(snpids = c("rs5050", "rs616488", "rs11249433",
                           "rs182799", "rs12565013", "rs11208590"),
                         genome.lib = "BSgenome.Hsapiens.UCSC.hg19",
                         snp.lib = "SNPlocs.Hsapiens.dbSNP.20120608",
                         half.window.size = 30,
                         default.par = TRUE,
                         mutation = FALSE)

## Warning:  the following rsids are not included in the database and discarded:
## rs182799
## Warning:  the following SNPs have more than 2 alleles, and only the first two alleles are used
as the SNP and the reference allele:
## rs12565013, rs11208590
## Warning:  the following sequences are discarded because the reference nucleotide matches to
neither a1 nor a2:
## snpid chr snp a1 a2
## rs11208590 chr1 41108149 A C
```

LoadSNPData may warn about the SNPs with inconsistent information and returns them in the output. The 'rsid.missing' output field captures SNPs that are not included in the *SNPlocs* package. The 'rsid.duplicate' output field captures SNPs with more than 2 alleles based on *SNPlocs* package. The 'rsid.rm' output field captures SNPs whose nucleotides in the reference genome do not match to either allele provided by the data source. SNPs in the 'rsid.missing' and 'rsid.rm' fields are discarded.

```
snp_info1$rsid.missing

## [1] "rs182799"

snp_info1$rsid.duplicate

## [1] "rs12565013" "rs11208590"

snp_info1$rsid.rm

## [1] "rs11208590"
```

### 3.2.3 Load SNP data through a pair of fasta files

Users can also provide SNP data through a pair of fasta files, one for the sequences around the SNP location for each allele. An example of such files is at http://pages.stat.wisc.edu/~keles/atSNP-Data/sample_1.fasta and http://pages.stat.wisc.edu/~keles/atSNP-Data/sample_2.fasta. We require that such a pair of fasta files must satisfy the following conditions:

1. All sequences from both files must be of the same odd number of length.
2. Sequences from the same position in each file are a pair of alleles. Their nucleotides must be the same except for the central nucleotide.

Such a pair of files can be loaded by function LoadFastaData:

```
snp_info2 <- LoadFastaData("http://pages.stat.wisc.edu/~keles/atSNP-Data/sample_1.fasta",
                           "http://pages.stat.wisc.edu/~keles/atSNP-Data/sample_2.fasta",
                           default.par = TRUE)
```

## 3.3 Affinity score tests

### 3.3.1 Load the example data

We use a toy example data set included in the package to introduce the usage of functions for affinity score tests.

```
data(example)
names(motif_library)

## [1] "SIX5_disc1" "MYC_disc1"

str(snpInfo)

## List of 5
##  $ sequence_matrix: int [1:61, 1:17] 4 3 1 4 3 2 2 1 3 3 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:17] "rs10910078" "rs4486391" "rs3748816" "rs2843401" ...
##  $ ref_base       : int [1:17] 4 1 1 4 4 4 4 1 2 2 ...
##  $ snp_base       : int [1:17] 2 4 3 2 2 2 2 2 4 4 ...
##  $ transition     : num [1:4, 1:4] 0.275 0.289 0.268 0.125 0.262 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "A" "C" "G" "T"
##   .. ..$ : chr [1:4] "A" "C" "G" "T"
##  $ prior          : Named num [1:4] 0.248 0.302 0.249 0.2
##   ..- attr(*, "names")= chr [1:4] "A" "C" "G" "T"

## to look at the motif information
data(encode_motif)

## Warning in data(encode_motif):  data set 'encode_motif' not found

encode_motifinfo[names(motif_library)]

##                                                SIX5_disc1
## "SIX5_GM12878_encode-Myers_seq_hsa_r1:MEME#1#Intergenic"
##                                                 MYC_disc1
## "USF2_K562_encode-Snyder_seq_hsa_r1:MDscan#1#Intergenic"
```

### 3.3.2 Compute affinity scores

The binding affinity scores for all pairs of SNP and PWM can be computed by the `ComputeMotifScore` function. It returns a list of two fields: 'snp.tbl' is a *data.table* containing the nucleotide sequences for each SNP; 'motif.scores' is a *data.table* containing the binding affinity scores for each SNP-motif pair.

```
  atsnp.scores <- ComputeMotifScore(motif_library, snpInfo, ncores = 2)
  head(atsnp.scores$snp.tbl)

##         snpid                                                        ref_seq
## 1: rs10910078 TGATGCCAGGTGGTCAGTGGGTTTTTGCCATCCGCCAGGAGCTTCACTGGGCCTCCCGTTG
## 2:  rs4486391 ATGGAGAATTCCACAGCTGATTGGAACCTAAACGAGAGAACCAAATGGACATCCCAGGGCT
## 3:  rs3748816 TTGGAGTACTCCTCGTCCAGGCGCCTGTTCATCTCCTCCAGGATGTAGTCAGGGTGCCCGA
## 4:  rs2843401 TCCTCCACCATTGTGCCAAACAGCGCCTGGTGGGGCCACCCGATCATCCCACGGGCCCCCA
## 5:  rs2843402 CACCTTCTGGGCTGCAGGACTTCCTGCCCTTTAGGAAAGGGAGGCAGCCCTTTCTTCCTCC
```

```
## 6:  rs2843403 CCCCCTAGGGCCTCCCTGCGGTTCCTTGTCTCCACCCTCACCCCAGCCCTGGAGCAGCCAC
##                                                                    snp_seq
## 1: TGATGCCAGGTGGTCAGTGGGTTTTTGCCACCCGCCAGGAGCTTCACTGGGCCTCCCGTTG
## 2: ATGGAGAATTCCACAGCTGATTGGAACCTATACGAGAGAACCAAATGGACATCCCAGGGCT
## 3: TTGGAGTACTCCTCGTCCAGGCGCCTGTTCGTCTCCTCCAGGATGTAGTCAGGGTGCCCGA
## 4: TCCTCCACCATTGTGCCAAACAGCGCCTGGCGGGGCCACCCGATCATCCCACGGGCCCCCA
## 5: CACCTTCTGGGCTGCAGGACTTCCTGCCCTCTAGGAAAGGGAGGCAGCCCTTTCTTCCTCC
## 6: CCCCCTAGGGCCTCCCTGCGGTTCCTTGTCCCCACCCTCACCCCAGCCCTGGAGCAGCCAC
##                                                                ref_seq_rev
## 1: CAACGGGAGGCCCAGTGAAGCTCCTGGCGGATGGCAAAAACCCACTGACCACCTGGCATCA
## 2: AGCCCTGGGATGTCCATTTGGTTCTCTCGTTTAGGTTCCAATCAGCTGTGGAATTCTCCAT
## 3: TCGGGCACCCTGACTACATCCTGGAGGAGATGAACAGGCGCCTGGACGAGGAGTACTCCAA
## 4: TGGGGGCCCGTGGGATGATCGGGTGGCCCCACCAGGCGCTGTTTGGCACAATGGTGGAGGA
## 5: GGAGGAAGAAAGGGCTGCCTCCCTTTCCTAAAGGGCAGGAAGTCCTGCAGCCCAGAAGGTG
## 6: GTGGCTGCTCCAGGGCTGGGGTGAGGGTGGAGACAAGGAACCGCAGGGAGGCCCTAGGGGG
##                                                                snp_seq_rev
## 1: CAACGGGAGGCCCAGTGAAGCTCCTGGCGGGTGGCAAAAACCCACTGACCACCTGGCATCA
## 2: AGCCCTGGGATGTCCATTTGGTTCTCTCGTATAGGTTCCAATCAGCTGTGGAATTCTCCAT
## 3: TCGGGCACCCTGACTACATCCTGGAGGAGACGAACAGGCGCCTGGACGAGGAGTACTCCAA
## 4: TGGGGGCCCGTGGGATGATCGGGTGGCCCCGCCAGGCGCTGTTTGGCACAATGGTGGAGGA
## 5: GGAGGAAGAAAGGGCTGCCTCCCTTTCCTAGAGGGCAGGAAGTCCTGCAGCCCAGAAGGTG
## 6: GTGGCTGCTCCAGGGCTGGGGTGAGGGTGGGGACAAGGAACCGCAGGGAGGCCCTAGGGGG
```

```
  head(atsnp.scores$motif.scores[, list(snpid, motif, log_lik_ref,
                                log_lik_snp, log_lik_ratio)])
```

```
##         snpid     motif log_lik_ref log_lik_snp log_lik_ratio
## 1: rs10910078 MYC_disc1   -95.57417   -92.79201    -2.7821535
## 2:  rs4486391 MYC_disc1   -94.37676   -79.51729   -14.8594729
## 3:  rs3748816 MYC_disc1   -96.67901   -99.39326     2.7142529
## 4:  rs2843401 MYC_disc1   -94.66127   -94.21702    -0.4442544
## 5:  rs2843402 MYC_disc1  -117.34142  -117.34142     0.0000000
## 6:  rs2843403 MYC_disc1  -115.81786  -115.81786     0.0000000
```

The affinity scores for the reference and the SNP alleles are represented by the 'log_lik_ref' and 'log_lik_snp' columns in '$motif.scores'. The affinity score change is included in the 'log_lik_ratio' column. These three affinity scores are tested in the subsequent steps. '$motif.scores' also include other columns for the position of the best matching subsequence on each allele. For a complete description on all these columns, users can look up the help documentation.

### 3.3.3 Compute p-values

After we have computed the binding affinity scores, they can be tested using the `ComputePValues` function. The result is a *data.table* extending the affinity score table by six columns:

- 'pval_ref': p-value for the reference allele affinity score.
- 'pval_snp': p-value for the SNP allele affinity score.
- 'pval_cond_ref' and 'pval_cond_snp': conditional p-values for the affinity scores of the reference and SNP alleles.
- 'pval_diff': p-value for the affinity score change between the two alleles.
- 'pval_rank': p-value for the rank test between the two alleles.

We recommend using 'pval_ref'and 'pval_snp' for assessing the significance of allele specific affinity; and using 'pval_rank' for assessing the significance of the SNP effect on the affinity change.

```
  atsnp.result <- ComputePValues(motif.lib = motif_library, snp.info = snpInfo,
                                 motif.scores = atsnp.scores$motif.scores,
                                 ncores = 2)
  head(atsnp.result[, list(snpid, motif, pval_ref, pval_snp, pval_rank, pval_diff)])
```

```
##         snpid     motif  pval_ref  pval_snp pval_rank pval_diff
## 1: rs10910078 MYC_disc1 0.5525082 0.3497673 0.4973136 0.5990576
```

```
## 2:  rs4486391 MYC_disc1 0.4246713 0.3248270 0.6181713 0.5319191
## 3:  rs3748816 MYC_disc1 0.6242812 0.7835283 0.6353814 0.6133480
## 4:  rs2843401 MYC_disc1 0.4709819 0.3901721 0.6629762 0.8090061
## 5:  rs2843402 MYC_disc1 0.9239229 0.9239229 1.0000000 1.0000000
## 6:  rs2843403 MYC_disc1 0.8407155 0.8407155 1.0000000 1.0000000
```

The *data.table* structure enables easy processing for prioritizing the SNP-PWM pairs based on the significance of affinity changes. We give a few examples here. First, we can sort this output table according to the 'pval_rank' column:

```
head(atsnp.result[order(pval_rank), list(snpid, motif, pval_ref, pval_snp, pval_rank)])
```

```
##         snpid      motif   pval_ref  pval_snp  pval_rank
## 1:  rs6667605 SIX5_disc1 0.06436599 0.7776495 0.01572935
## 2:  rs2843401 SIX5_disc1 0.08081194 0.2301178 0.01654064
## 3:  rs6424092  MYC_disc1 0.05381326 0.2888447 0.02528970
## 4:  rs2985857  MYC_disc1 0.09264310 0.3248270 0.14064803
## 5:  rs2843402 SIX5_disc1 0.13659677 0.3283295 0.14789201
## 6: rs10797432 SIX5_disc1 0.33296184 0.1460646 0.17415016
```

Second, we can also select the SNP-PWM pairs subject to a threshold in 'pval_rank':

```
atsnp.result[pval_rank <= 0.1, list(snpid, motif, pval_ref, pval_snp, pval_rank)]
```

```
##        snpid      motif   pval_ref  pval_snp  pval_rank
## 1: rs2843401 SIX5_disc1 0.08081194 0.2301178 0.01654064
## 2: rs6424092  MYC_disc1 0.05381326 0.2888447 0.02528970
## 3: rs6667605 SIX5_disc1 0.06436599 0.7776495 0.01572935
```

### 3.3.4 Multiple testing adjustment

We can apply multiple testing adjustment to the p-values. *atSNP* does not implement any multiple testing adjustment internally. Users have the flexibility of choosing the adjustment method based on their specific application. For example, if we want to adjust 'pval_rank' from all pairs of SNP-PWM pairs using the Benjamini-Hochberg's procedure, we may compute:

```
atsnp.result[, pval_rank_bh := p.adjust(pval_rank, method = "BH")]
```

```
##          snpid      motif  pval_rank pval_rank_bh
##  1: rs10910078  MYC_disc1 0.49731358    0.9066472
##  2:  rs4486391  MYC_disc1 0.61817128    0.9066472
##  3:  rs3748816  MYC_disc1 0.63538138    0.9066472
##  4:  rs2843401  MYC_disc1 0.66297624    0.9066472
##  5:  rs2843402  MYC_disc1 1.00000000    1.0000000
##  6:  rs2843403  MYC_disc1 1.00000000    1.0000000
##  7:  rs2843404  MYC_disc1 0.33680154    0.9066472
##  8:  rs2985855  MYC_disc1 0.69437460    0.9066472
##  9: rs10910078 SIX5_disc1 0.45687986    0.9066472
## 10:  rs4486391 SIX5_disc1 1.00000000    1.0000000
## 11:  rs3748816 SIX5_disc1 0.62512531    0.9066472
## 12:  rs2843401 SIX5_disc1 0.01654064    0.2811909
## 13:  rs2843402 SIX5_disc1 0.14789201    0.9066472
## 14:  rs2843403 SIX5_disc1 0.57145169    0.9066472
## 15:  rs2843404 SIX5_disc1 0.48140291    0.9066472
## 16:  rs2985855 SIX5_disc1 1.00000000    1.0000000
## 17:  rs2296442  MYC_disc1 0.29890362    0.9066472
## 18: rs10797432  MYC_disc1 0.64089998    0.9066472
```

```
## 19:  rs6667605  MYC_disc1 0.74665067    0.9066472
## 20:  rs4648648  MYC_disc1 0.71484150    0.9066472
## 21:   rs734999  MYC_disc1 0.73348109    0.9066472
## 22:  rs2764845  MYC_disc1 0.72498353    0.9066472
## 23:  rs2764841  MYC_disc1 0.73131633    0.9066472
## 24:  rs2985857  MYC_disc1 0.14064803    0.9066472
## 25:  rs6424092  MYC_disc1 0.02528970    0.2866166
## 26:  rs2296442 SIX5_disc1 0.30722405    0.9066472
## 27: rs10797432 SIX5_disc1 0.17415016    0.9066472
## 28:  rs6667605 SIX5_disc1 0.01572935    0.2811909
## 29:  rs4648648 SIX5_disc1 0.26849187    0.9066472
## 30:   rs734999 SIX5_disc1 0.21057043    0.9066472
## 31:  rs2764845 SIX5_disc1 0.93486937    1.0000000
## 32:  rs2764841 SIX5_disc1 0.53680487    0.9066472
## 33:  rs2985857 SIX5_disc1 1.00000000    1.0000000
## 34:  rs6424092 SIX5_disc1 0.73416105    0.9066472
##          snpid      motif  pval_rank pval_rank_bh
```

Alternatively, if we want to compute Storey's q-values, we may utilize the *qvalue* package from *Bioconductor*:

```
library(qvalue)
atsnp.result[, qval_rank := qvalue(pval_rank)$qvalues]
```

Rather than adjusting all the p-values, if we have a large list of SNPs, we can adjust the p-values for each motif separately. This can be done by adding a 'by=...' argument in the `data.table`:

```
atsnp.result[, pval_rank_bh := p.adjust(pval_rank, method = "BH"), by = motif]
atsnp.result[, qval_rank := qvalue(pval_rank)$qvalues, by = motif]
```

Similarly, if we have a large motif library, we may also adjust the p-values for each SNP:

```
atsnp.result[, pval_rank_bh := p.adjust(pval_rank, method = "BH"), by = snpid]
atsnp.result[, qval_rank := qvalue(pval_rank)$qvalues, by = snpid]
```

## 3.4 Additional analysis

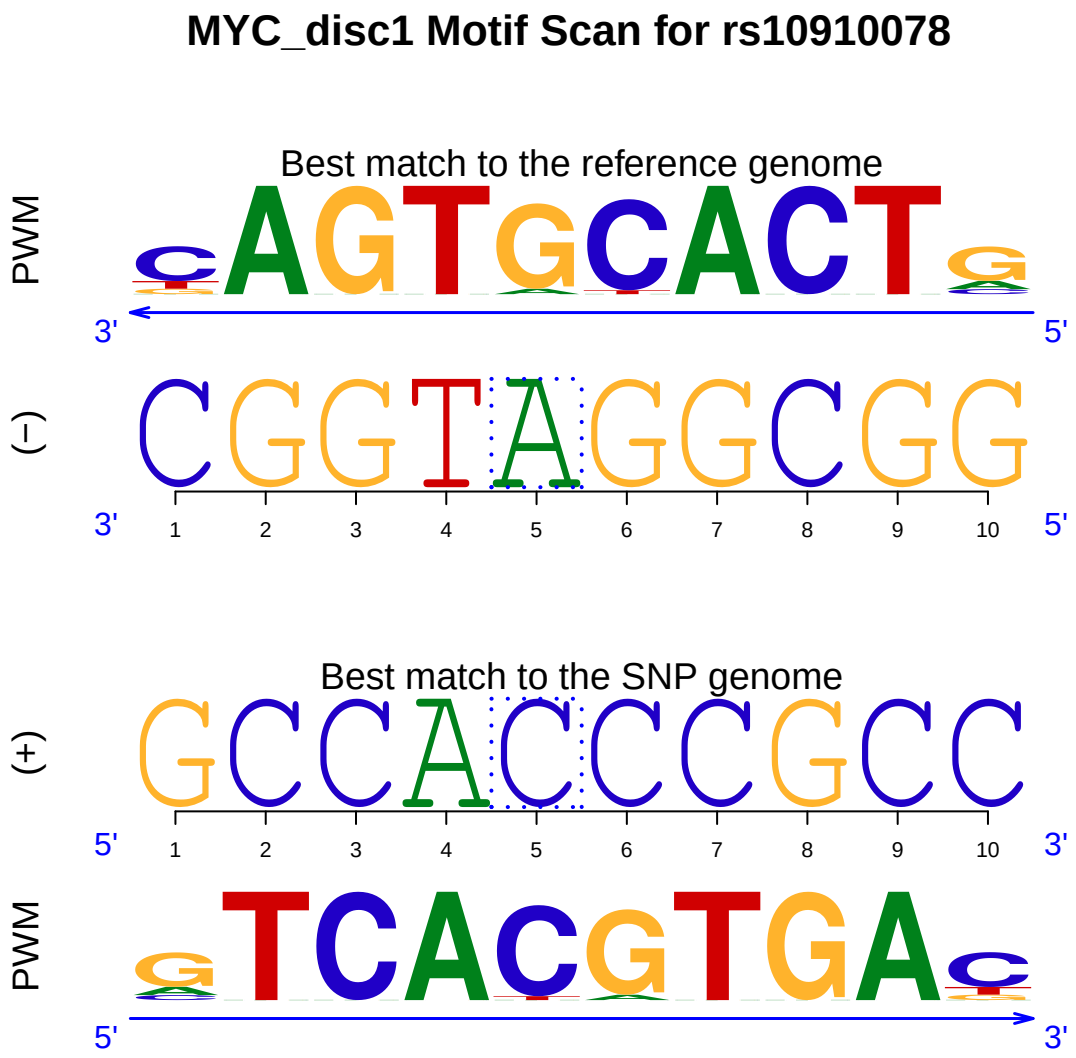
atSNP provides additional functions to extract the matched nucleotide subsequences that match to the motifs. Function `MatchSubsequence` adds the subsequence matches to the affinity score table by using the motif library and the SNP set. The subsequences matching to the motif in the two alleles are returned in the 'ref_match_seq' and 'snp_match_seq' columns. The 'IUPAC' column returns the IUPAC letters of the motifs. Notice that if you have a large number of SNPs and motifs, the returned table can be very large.

```
  match_result <- MatchSubsequence(snp.tbl = atsnp.scores$snp.tbl,
                                   motif.scores = atsnp.result,
                                   motif.lib = motif_library,
                                   snpids = c("rs10910078", "rs4486391"),
                                   motifs = names(motif_library)[1:2],
                                   ncores = 2)
  match_result[, list(snpid, motif, IUPAC, ref_match_seq, snp_match_seq)]
```

```
##         snpid      motif      IUPAC ref_match_seq snp_match_seq
## 1: rs10910078  MYC_disc1 GTCACGTGAC    GGCGGATGGC    GCCACCCGCC
## 2: rs10910078 SIX5_disc1 GARWTGTAGT    CTGGCGGATG    GGCGGGTGGC
## 3:  rs4486391  MYC_disc1 GTCACGTGAC    TAAACGAGAG    CTCGTATAGG
## 4:  rs4486391 SIX5_disc1 GARWTGTAGT    CTCGTTTAGG    CTCGTATAGG
```

To visualize how each motif is matched to each allele using the `plotMotifMatch` function:

```
plotMotifMatch(snp.tbl = atsnp.scores$snp.tbl,
               motif.scores = atsnp.scores$motif.scores,
               snpid = atsnp.scores$snp.tbl$snpid[1],
               motif.lib = motif_library,
               motif = atsnp.scores$motif.scores$motif[1])
```




## 4 Session Information

```
## R version 3.1.1 (2014-07-10)
## Platform: x86_64-redhat-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=zh_TW.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=en_US.UTF-8     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] grid      parallel  stats     graphics  grDevices utils     datasets  methods
```

```
## [9] base
##
## other attached packages:
##  [1] SNPlocs.Hsapiens.dbSNP.20120608_0.99.9 atSNP_1.0
##  [3] motifStack_1.6.5                       ade4_1.6-2
##  [5] MotIV_1.18.0                           grImport_0.9-0
##  [7] XML_3.98-1.1                           doParallel_1.0.8
##  [9] iterators_1.0.7                        foreach_1.4.2
## [11] data.table_1.9.4                       Rcpp_0.11.4
## [13] BSgenome.Hsapiens.UCSC.hg19_1.3.19     BSgenome_1.30.0
## [15] Biostrings_2.30.1                      GenomicRanges_1.14.4
## [17] XVector_0.2.0                          IRanges_1.20.7
## [19] BiocGenerics_0.8.0                     BiocInstaller_1.12.1
##
## loaded via a namespace (and not attached):
##  [1] BiocStyle_1.0.0 chron_2.3-45    codetools_0.2-8 compiler_3.1.1  evaluate_0.5.5
##  [6] formatR_1.0     highr_0.4       knitr_1.9       lattice_0.20-29 plyr_1.8.1
## [11] reshape2_1.4.1  rGADEM_2.10.0   seqLogo_1.28.0  stats4_3.1.1    stringr_0.6.2
## [16] tools_3.1.1
```

## References


[1] Hock Peng Chan, Nancy Ruonan Zhang, and Louis H.Y. Chen. Importance sampling of word patterns in DNA and protein sequences. *Journal of Computational Biology*, 17(12):1697–1709, 2010.

[2] Charles E. Grant, Timothy L. Bailey, and William Stafford Nobel. FIMO: Scanning for occurrences of a given motif. *Bioinformatics*, 7:1017, 2011.

[3] L.A. Hindorff, J. MacArthur J, J. Morales, H.A. Junkins, P.N. Hall, A.K. Klemm, and T.A. Manolio. A catalog of published genome-wide association studies.

[4] Geoff Macintyre, James Bailey, Izhak Haviv, and Adam Kowalczyk. is-rSNP: a novel technique for in silico regulatory SNP detection. *Bioinformatics*, 26(18):524–530, 2010.

[5] Chandler Zuo, Sunyoung Shin, and Sunduz Keles. atsnp: affinity tests for regulatory snp detection. *Bioinformatics (submitted)*, 2015.