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.
- *doMC* is used for parallel computation.
- *Rcpp* interfaces the C++ codes that implements the importance sampling algorithm.

In addition, users also need to install the annotation package 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. To install it from the *Bioconductor* repository,

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

Notice that the annotation package is usually large and this installation step may take a substantial amout of time.

3 Example

3.1 Load motif and SNP data

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:  doMC
## 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_motif)
length(motif_encode)

## [1] 2065

motif_encode[seq(3)]

## $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
##
## $MYC_disc1
##              [,1]        [,2]        [,3]        [,4]
##  [1,] 1.73516e-01 1.05023e-01 7.21461e-01 1.00000e-10
##  [2,] 1.00000e-10 1.00000e-10 1.00000e-10 1.00000e+00
##  [3,] 1.00000e-10 1.00000e+00 1.00000e-10 1.00000e-10
##  [4,] 1.00000e+00 1.00000e-10 1.00000e-10 1.00000e-10
##  [5,] 1.00000e-10 9.58904e-01 1.00000e-10 4.10960e-02
##  [6,] 5.93610e-02 1.00000e-10 9.40639e-01 1.00000e-10
##  [7,] 1.00000e-10 1.00000e-10 1.00000e-10 1.00000e+00
##  [8,] 1.00000e-10 1.00000e-10 1.00000e+00 1.00000e-10
##  [9,] 1.00000e+00 1.00000e-10 1.00000e-10 1.00000e-10
## [10,] 1.00000e-10 7.26028e-01 1.14155e-01 1.59817e-01
##
## $SRF_disc1
##              [,1]  [,2]  [,3]        [,4]
##  [1,] 1.00000e-10 1e+00 1e-10 1.00000e-10
##  [2,] 1.00000e-10 1e+00 1e-10 1.00000e-10
##  [3,] 4.95495e-01 1e-10 1e-10 5.04505e-01
##  [4,] 2.61261e-01 1e-10 1e-10 7.38739e-01
##  [5,] 1.00000e+00 1e-10 1e-10 1.00000e-10
##  [6,] 1.00000e-10 1e-10 1e-10 1.00000e+00
##  [7,] 7.29730e-01 1e-10 1e-10 2.70270e-01
##  [8,] 5.04505e-01 1e-10 1e-10 4.95495e-01
##  [9,] 1.00000e-10 1e-10 1e+00 1.00000e-10
## [10,] 1.00000e-10 1e-10 1e+00 1.00000e-10
```

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

```
motif_encode[[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(motif_encode[[1]])

## [1] "GARWTGTAGT"
```

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

```
length(motif_info)

## [1] 2065

head(motif_info)

##                                                  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. `motif_info` allows easy looking up the motif information for a specific PWM. For example, to look up the motif information for the first PWM in `motif_encode`:

```
motif_info[names(motif_encode[1])]

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

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.

```
pwms <- LoadMotifLibrary(
 "http://meme.nbcr.net/meme/examples/sample-dna-motif.meme-io")
pwms <- LoadMotifLibrary(
 "http://compbio.mit.edu/encode-motifs/motifs.txt",
 tag = ">", transpose = FALSE, field = 1,
 sep = c("\t", " ", ">"), skipcols = 1,
 skiprows = 1, pseudocount = 0)
pwms <- LoadMotifLibrary(
 "http://johnsonlab.ucsf.edu/mochi_files/JASPAR_motifs_H_sapiens.txt",
 tag = "/NAME",skiprows = 1, skipcols = 0, transpose = FALSE,
 field = 2)
pwms <- LoadMotifLibrary(
 "http://jaspar.genereg.net/html/DOWNLOAD/ARCHIVE/JASPAR2010/all_data/matrix_only/matrix.txt",
 tag = ">", skiprows = 1, skipcols = 1, transpose = TRUE,
 field = 1, sep = c("\t", " ", "\\[", "\\]", ">"),
 pseudocount = 1)
pwms <- LoadMotifLibrary(
 "http://jaspar.genereg.net/html/DOWNLOAD/JASPAR_CORE/pfm/nonredundant/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")
```

The data set for the SNP information must be a table including 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)

## 3 sequences are discarded because the reference nucleotide matches to neither a1 nor a2.

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

## [1] FALSE
```

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. '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. 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
```

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.

A toy sample data set including a preloaded motif library and a SNP set is included in the package:

```
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)
motif_info[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"
```

3.2 Affinity score tests

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.

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

##          snpid                                                       ref_seq
##  1: rs10910078 TGATGCCAGGTGGTCAGTGGGTTTTTGCCATCCGCCAGGAGCTTCACTGGGCCTCCCGTTG
##  2:  rs4486391 ATGGAGAATTCCACAGCTGATTGGAACCTAAACGAGAGAACCAAATGGACATCCCAGGGCT
##  3:  rs3748816 TTGGAGTACTCCTCGTCCAGGCGCCTGTTCATCTCCTCCAGGATGTAGTCAGGGTGCCCGA
##  4:  rs2843401 TCCTCCACCATTGTGCCAAACAGCGCCTGGTGGGGCCACCCGATCATCCCACGGGCCCCCA
##  5:  rs2843402 CACCTTCTGGGCTGCAGGACTTCCTGCCCTTTAGGAAAGGGAGGCAGCCCTTTCTTCCTCC
##  6:  rs2843403 CCCCCTAGGGCCTCCCTGCGGTTCCTTGTCTCCACCCTCACCCCAGCCCTGGAGCAGCCAC
##  7:  rs2843404 AAATGGAATATTTAATTTGAAACTTTCCAATAAAGAAATTTCCAGACCCATTTGGCTTCAC
##  8:  rs2985855 ACCTGATAAAGGAAATGTATGAAGCAGCAGAAGCAACAAAAACAACTCCATAGCAAACATA
##  9:  rs2296442 CCGCTTCCTCGTCTGGGACCACGATCCCATCGGGCGTGACTTCATTGGCCAGAGGACGCTG
## 10: rs10797432 GACTCACAGGTGGGAGACAGGAGTTCCGACCGCCAGGGGGAGAGTCCTGGAGGATCCTGGG
## 11:  rs6667605 TCCCACAAATGCAGAAAGCTCAACAGACCCCAAGAGGGGTAAATAGAGAGGCATGCACTGC
## 12:  rs4648648 CAGGTCCTGCGATCTCCCCACGCCCTGACAGTGACCTATCTTTGTGCACACACGTGTGTTT
## 13:   rs734999 CCACTGAAATACCCGTGGGAAAGAAAAGCACAACAGAGAACAGGAGACTTATGTGACTCCG
## 14:  rs2764845 CACCATGGCCAAGCCTGTCACCTCACCTGGGTGACCACATCGGCCTCCATGCTGACCCCGC
## 15:  rs2764841 CTGTTTCTGCTCCCGGGAAATCACCCCGCCGCCTCTTCAGGCCTTTAAGGTCTCAAATGTC
## 16:  rs2985857 CACTCTTGAAGAACAAAGTTGAAATATATACTCTATTGACTATCAAGACATTATAAAGCTG
## 17:  rs6424092 ATCTCACTGTCCATTAAAAAAATCAACTCACAGTAGATTGTAGACCTAAGCAAACCTGAGG
##                                                                  snp_seq
##  1: TGATGCCAGGTGGTCAGTGGGTTTTTGCCACCCGCCAGGAGCTTCACTGGGCCTCCCGTTG
##  2: ATGGAGAATTCCACAGCTGATTGGAACCTATACGAGAGAACCAAATGGACATCCCAGGGCT
##  3: TTGGAGTACTCCTCGTCCAGGCGCCTGTTCGTCTCCTCCAGGATGTAGTCAGGGTGCCCGA
##  4: TCCTCCACCATTGTGCCAAACAGCGCCTGGCGGGGCCACCCGATCATCCCACGGGCCCCCA
##  5: CACCTTCTGGGCTGCAGGACTTCCTGCCCTCTAGGAAAGGGAGGCAGCCCTTTCTTCCTCC
##  6: CCCCCTAGGGCCTCCCTGCGGTTCCTTGTCCCCACCCTCACCCCAGCCCTGGAGCAGCCAC
##  7: AAATGGAATATTTAATTTGAAACTTTCCAACAAAGAAATTTCCAGACCCATTTGGCTTCAC
##  8: ACCTGATAAAGGAAATGTATGAAGCAGCAGCAGCAACAAAAACAACTCCATAGCAAACATA
##  9: CCGCTTCCTCGTCTGGGACCACGATCCCATTGGGCGTGACTTCATTGGCCAGAGGACGCTG
## 10: GACTCACAGGTGGGAGACAGGAGTTCCGACTGCCAGGGGGAGAGTCCTGGAGGATCCTGGG
## 11: TCCCACAAATGCAGAAAGCTCAACAGACCCTAAGAGGGGTAAATAGAGAGGCATGCACTGC
## 12: CAGGTCCTGCGATCTCCCCACGCCCTGACAATGACCTATCTTTGTGCACACACGTGTGTTT
## 13: CCACTGAAATACCCGTGGGAAAGAAAAGCATAACAGAGAACAGGAGACTTATGTGACTCCG
## 14: CACCATGGCCAAGCCTGTCACCTCACCTGGTTGACCACATCGGCCTCCATGCTGACCCCGC
## 15: CTGTTTCTGCTCCCGGGAAATCACCCCGCCACCTCTTCAGGCCTTTAAGGTCTCAAATGTC
## 16: CACTCTTGAAGAACAAAGTTGAAATATATATTCTATTGACTATCAAGACATTATAAAGCTG
## 17: ATCTCACTGTCCATTAAAAAAATCAACTCAAAGTAGATTGTAGACCTAAGCAAACCTGAGG
##                                                              ref_seq_rev
##  1: CAACGGGAGGCCCAGTGAAGCTCCTGGCGGATGGCAAAAACCCACTGACCACCTGGCATCA
##  2: AGCCCTGGGATGTCCATTTGGTTCTCTCGTTTAGGTTCCAATCAGCTGTGGAATTCTCCAT
##  3: TCGGGCACCCTGACTACATCCTGGAGGAGATGAACAGGCGCCTGGACGAGGAGTACTCCAA
##  4: TGGGGGCCCGTGGGATGATCGGGTGGCCCCACCAGGCGCTGTTTGGCACAATGGTGGAGGA
##  5: GGAGGAAGAAAGGGCTGCCTCCCTTTCCTAAAGGGCAGGAAGTCCTGCAGCCCAGAAGGTG
##  6: GTGGCTGCTCCAGGGCTGGGGTGAGGGTGGAGACAAGGAACCGCAGGGAGGCCCTAGGGGG
##  7: GTGAAGCCAAATGGGTCTGGAAATTTCTTTATTGGAAAGTTTCAAATTAAATATTCCATTT
##  8: TATGTTTGCTATGGAGTTGTTTTTGTTGCTTCTGCTGCTTCATACATTTCCTTTATCAGGT
##  9: CAGCGTCCTCTGGCCAATGAAGTCACGCCCGATGGGATCGTGGTCCCAGACGAGGAAGCGG
## 10: CCCAGGATCCTCCAGGACTCTCCCCCTGGCGGTCGGAACTCCTGTCTCCCACCTGTGAGTC
## 11: GCAGTGCATGCCTCTCTATTTACCCCTCTTGGGGTCTGTTGAGCTTTCTGCATTTGTGGGA
## 12: AAACACACGTGTGTGCACAAAGATAGGTCACTGTCAGGGCGTGGGGAGATCGCAGGACCTG
## 13: CGGAGTCACATAAGTCTCCTGTTCTCTGTTGTGCTTTTCTTTCCCACGGGTATTTCAGTGG
## 14: GCGGGGTCAGCATGGAGGCCGATGTGGTCACCCAGGTGAGGTGACAGGCTTGGCCATGGTG
## 15: GACATTTGAGACCTTAAAGGCCTGAAGAGGCGGCGGGGTGATTTCCCGGGAGCAGAAACAG
## 16: CAGCTTTATAATGTCTTGATAGTCAATAGAGTATATATTTCAACTTTGTTCTTCAAGAGTG
## 17: CCTCAGGTTTGCTTAGGTCTACAATCTACTGTGAGTTGATTTTTTTAATGGACAGTGAGAT
##                                                              snp_seq_rev
##  1: CAACGGGAGGCCCAGTGAAGCTCCTGGCGGGTGGCAAAAACCCACTGACCACCTGGCATCA
##  2: AGCCCTGGGATGTCCATTTGGTTCTCTCGTATAGGTTCCAATCAGCTGTGGAATTCTCCAT
##  3: TCGGGCACCCTGACTACATCCTGGAGGAGACGAACAGGCGCCTGGACGAGGAGTACTCCAA
##  4: TGGGGGCCCGTGGGATGATCGGGTGGCCCCGCCAGGCGCTGTTTGGCACAATGGTGGAGGA
##  5: GGAGGAAGAAAGGGCTGCCTCCCTTTCCTAGAGGGCAGGAAGTCCTGCAGCCCAGAAGGTG
##  6: GTGGCTGCTCCAGGGCTGGGGTGAGGGTGGGGACAAGGAACCGCAGGGAGGCCCTAGGGGG
##  7: GTGAAGCCAAATGGGTCTGGAAATTTCTTTGTTGGAAAGTTTCAAATTAAATATTCCATTT
##  8: TATGTTTGCTATGGAGTTGTTTTTGTTGCTGCTGCTGCTTCATACATTTCCTTTATCAGGT
##  9: CAGCGTCCTCTGGCCAATGAAGTCACGCCCAATGGGATCGTGGTCCCAGACGAGGAAGCGG
## 10: CCCAGGATCCTCCAGGACTCTCCCCCTGGCAGTCGGAACTCCTGTCTCCCACCTGTGAGTC
## 11: GCAGTGCATGCCTCTCTATTTACCCCTCTTAGGGTCTGTTGAGCTTTCTGCATTTGTGGGA
## 12: AAACACACGTGTGTGCACAAAGATAGGTCATTGTCAGGGCGTGGGGAGATCGCAGGACCTG
## 13: CGGAGTCACATAAGTCTCCTGTTCTCTGTTATGCTTTTCTTTCCCACGGGTATTTCAGTGG
## 14: GCGGGGTCAGCATGGAGGCCGATGTGGTCAACCAGGTGAGGTGACAGGCTTGGCCATGGTG
## 15: GACATTTGAGACCTTAAAGGCCTGAAGAGGTGGCGGGGTGATTTCCCGGGAGCAGAAACAG
## 16: CAGCTTTATAATGTCTTGATAGTCAATAGAATATATATTTCAACTTTGTTCTTCAAGAGTG
## 17: CCTCAGGTTTGCTTAGGTCTACAATCTACTTTGAGTTGATTTTTTTAATGGACAGTGAGAT
```

```
 motif_score$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
##  7:  rs2843404  MYC_disc1   -95.73058  -118.75643    23.0258509
##  8:  rs2985855  MYC_disc1  -116.88074  -120.49201     3.6112717
##  9: rs10910078 SIX5_disc1   -46.06943   -38.82055    -7.2488780
## 10:  rs4486391 SIX5_disc1   -41.21034   -41.21034     0.0000000
## 11:  rs3748816 SIX5_disc1   -51.99542   -40.50007   -11.4953572
## 12:  rs2843401 SIX5_disc1   -19.33735   -23.09387     3.7565188
## 13:  rs2843402 SIX5_disc1   -20.74899   -23.90834     3.1593576
## 14:  rs2843403 SIX5_disc1   -38.21561   -41.13338     2.9177676
## 15:  rs2843404 SIX5_disc1   -43.06925   -38.31571    -4.7535476
## 16:  rs2985855 SIX5_disc1   -58.35713   -58.35713     0.0000000
## 17:  rs2296442  MYC_disc1   -71.67739   -75.86532     4.1879305
## 18: rs10797432  MYC_disc1  -117.72909   -99.39326   -18.3358300
## 19:  rs6667605  MYC_disc1   -75.86532   -79.01520     3.1498802
## 20:  rs4648648  MYC_disc1   -92.79201   -94.21702     1.4250085
## 21:   rs734999  MYC_disc1   -75.86532   -79.01520     3.1498802
## 22:  rs2764845  MYC_disc1   -50.51744   -50.03458    -0.4828589
## 23:  rs2764841  MYC_disc1   -94.64204   -94.23625    -0.4057914
## 24:  rs2985857  MYC_disc1   -71.57423   -92.42357    20.8493426
## 25:  rs6424092  MYC_disc1   -53.34156   -76.32544    22.9838866
## 26:  rs2296442 SIX5_disc1   -23.43206   -21.51514    -1.9169281
## 27: rs10797432 SIX5_disc1   -24.19256   -21.27479    -2.9177676
## 28:  rs6667605 SIX5_disc1   -18.25728   -41.28313    23.0258509
## 29:  rs4648648 SIX5_disc1   -44.20012   -36.50922    -7.6909014
## 30:   rs734999 SIX5_disc1   -36.30181   -56.49106    20.1892485
## 31:  rs2764845 SIX5_disc1   -41.43849   -41.69843     0.2599394
## 32:  rs2764841 SIX5_disc1   -17.46570   -16.16641    -1.2992901
## 33:  rs2985857 SIX5_disc1   -54.58632   -54.58632     0.0000000
## 34:  rs6424092 SIX5_disc1   -38.05012   -37.46235    -0.5877710
##          snpid      motif log_lik_ref log_lik_snp log_lik_ratio
```

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.

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.

```
  motif.scores <- ComputePValues(motif.lib = motif_library, snp.info = snpInfo,
                                motif.scores = motif_scores$motif.scores,
                                ncores = 2)
  motif.scores[, 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 0.55250823 0.34976728 0.49731358 0.599057634
##  2:  rs4486391  MYC_disc1 0.42467126 0.32482700 0.61817128 0.531919133
##  3:  rs3748816  MYC_disc1 0.62428125 0.78352831 0.63538138 0.613347967
##  4:  rs2843401  MYC_disc1 0.47098190 0.39017206 0.66297624 0.809006055
##  5:  rs2843402  MYC_disc1 0.92392287 0.92392287 1.00000000 1.000000000
##  6:  rs2843403  MYC_disc1 0.84071552 0.84071552 1.00000000 1.000000000
##  7:  rs2843404  MYC_disc1 0.56662380 0.97929975 0.33680154 0.150135756
##  8:  rs2985855  MYC_disc1 0.85727211 0.99400000 0.69437460 0.572437051
##  9: rs10910078 SIX5_disc1 0.87876479 0.62172720 0.45687986 0.280402306
## 10:  rs4486391 SIX5_disc1 0.77460249 0.77460249 1.00000000 1.000000000
## 11:  rs3748816 SIX5_disc1 0.90275130 0.74334730 0.62512531 0.206882124
## 12:  rs2843401 SIX5_disc1 0.08081194 0.23011782 0.01654064 0.434515202
## 13:  rs2843402 SIX5_disc1 0.13659677 0.32832950 0.14789201 0.490855902
## 14:  rs2843403 SIX5_disc1 0.60110887 0.77253940 0.57145169 0.618352633
## 15:  rs2843404 SIX5_disc1 0.84323561 0.61067088 0.48140291 0.354767558
## 16:  rs2985855 SIX5_disc1 0.97623554 0.97623554 1.00000000 1.000000000
## 17:  rs2296442  MYC_disc1 0.11261743 0.26841651 0.29890362 0.553813438
## 18: rs10797432  MYC_disc1 0.95726548 0.78177027 0.64089998 0.457487238
## 19:  rs6667605  MYC_disc1 0.27303296 0.29517812 0.74665067 0.592775953
## 20:  rs4648648  MYC_disc1 0.33416921 0.37746258 0.71484150 0.716330202
## 21:   rs734999  MYC_disc1 0.26841651 0.29517812 0.73348109 0.591312146
## 22:  rs2764845  MYC_disc1 0.02476245 0.02225671 0.72498353 0.803576225
## 23:  rs2764841  MYC_disc1 0.44985448 0.40610585 0.73131633 0.816179825
## 24:  rs2985857  MYC_disc1 0.09264310 0.32482700 0.14064803 0.367260674
## 25:  rs6424092  MYC_disc1 0.05381326 0.28884471 0.02528970 0.171276172
## 26:  rs2296442 SIX5_disc1 0.30747596 0.17194308 0.30722405 0.688903343
## 27: rs10797432 SIX5_disc1 0.33296184 0.14606464 0.17415016 0.621522278
## 28:  rs6667605 SIX5_disc1 0.06436599 0.77764949 0.01572935 0.002994372
## 29:  rs4648648 SIX5_disc1 0.87216144 0.48674767 0.26849187 0.268226489
## 30:   rs734999 SIX5_disc1 0.44673064 0.95825787 0.21057043 0.102711064
## 31:  rs2764845 SIX5_disc1 0.80012831 0.80656654 0.93486937 0.915061559
## 32:  rs2764841 SIX5_disc1 0.05984278 0.04457300 0.53680487 0.751367323
## 33:  rs2985857 SIX5_disc1 0.92853988 0.92853988 1.00000000 1.000000000
## 34:  rs6424092 SIX5_disc1 0.59650110 0.54347923 0.73416105 0.866256977
##           snpid      motif   pval_ref   pval_snp  pval_rank   pval_diff
```

3.3 Additional analysis

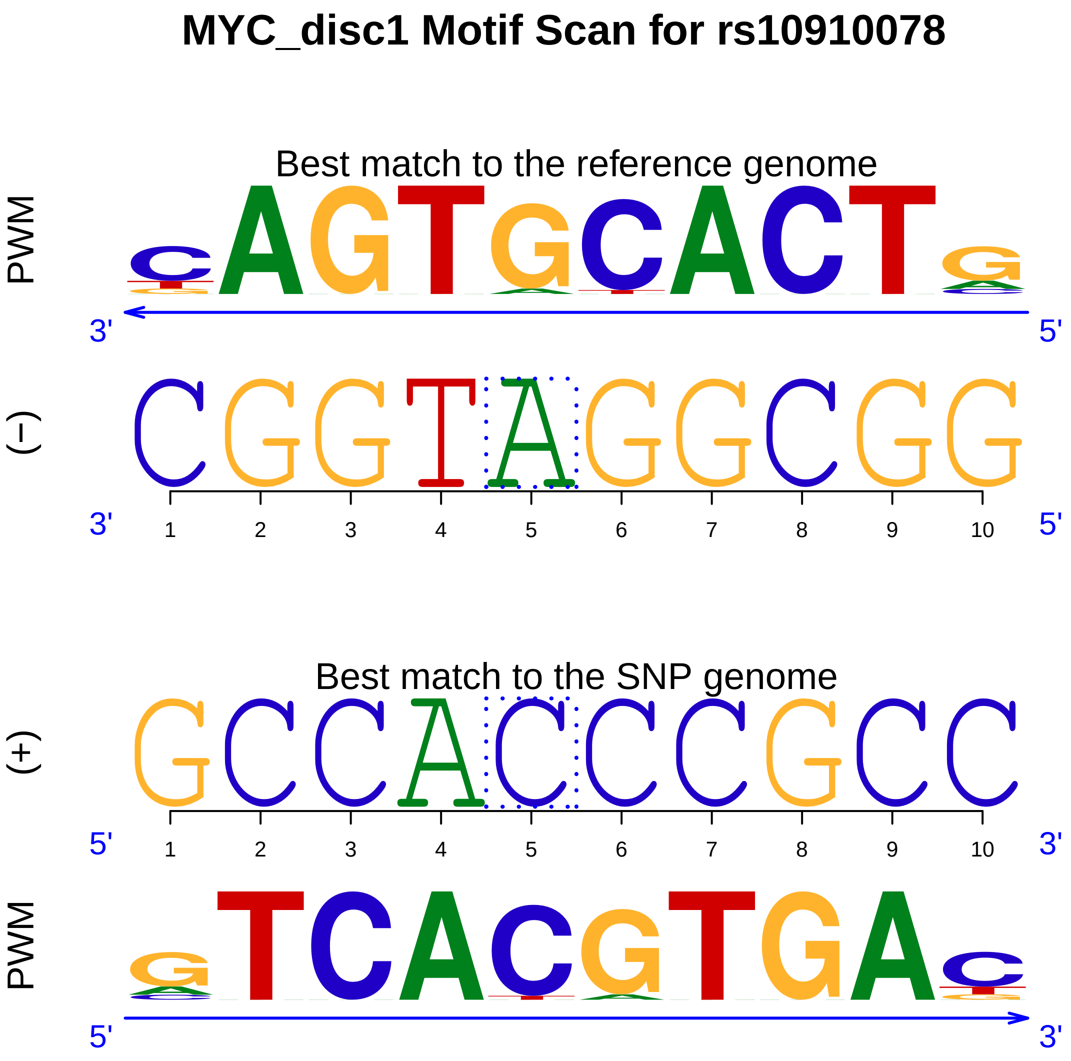
atSNP provides additional functions to extract the matched nucleotide subsequences that match to the motifs. Function `MatchSubsequence` adds the subsequence matches to the affinity score table by using the motif library and the SNP set. The subsequences matching to the motif in the two alleles are returned 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 = motif_scores$snp.tbl,
                                 motif.scores = motif.scores,
                                 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 = motif_scores$snp.tbl,
               motif.scores = motif_scores$motif.scores,
               snpid = motif_scores$snp.tbl$snpid[1],
               motif.lib = motif_library,
               motif = motif_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] atSNP_1.0                        motifStack_1.6.5
##  [3] ade4_1.6-2                       MotIV_1.18.0
##  [5] grImport_0.9-0                   XML_3.98-1.1
##  [7] doMC_1.3.3                         iterators_1.0.7
##  [9] foreach_1.4.2                      data.table_1.9.4
## [11] Rcpp_0.11.4                        BSgenome.Hsapiens.UCSC.hg19_1.3.19
## [13] BSgenome_1.30.0                    Biostrings_2.30.1
## [15] GenomicRanges_1.14.4               XVector_0.2.0
## [17] IRanges_1.20.7                     BiocGenerics_0.8.0
## [19] 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.