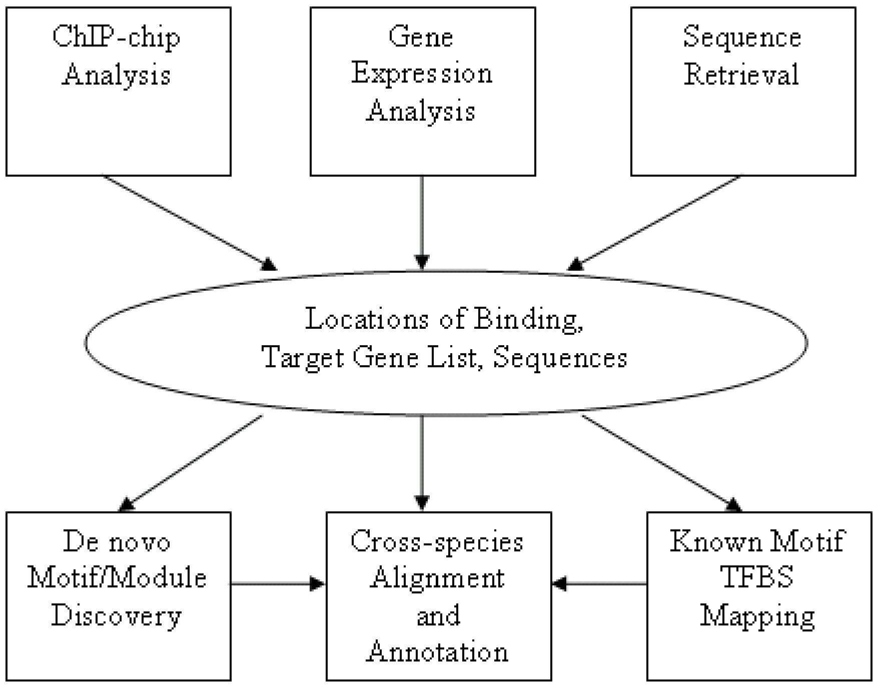
CisGenome User's Manual

1. Overview

1.1 Basic Framework of CisGenome

CisGenome is a collection of stand-alone programs that support cis-regulatory analysis in mammalian genomes. Functions provided by the software include gene selection from microarrays, peak detection from ChIP-chip tiling arrays, genomic sequence retrieval, de novo motif and module discovery, transcription factor binding site (TFBS) mapping, cross-species alignments and annotation using TFBSs. The basic structure of this computational toolbox is shown below.


ChIP-chip Analysis
Gene Expression Analysis
Sequence Retrieval
Locations of Binding, Target Gene List, Sequences
De novo Motif/Module Discovery
Cross-species Alignment and Annotation
Known Motif TFBS Mapping

A crucial feature of CisGenome is its ability to support genome-scale computational analysis in a customized fashion. Unlike most web servers that return analysis results in a predefined form, the wide spectrum of CisGenome functions allows users to design the analysis procedure best suited for their own purpose.

1.2. Installation

CisGenome is implemented in ANSI C and can be used on both MS Windows and Unix.

1.2.1 Installation on Windows

To install CisGenome on Windows, one can download the executable files from http://biogibbs.stanford.edu/~jihk/CisGenome/index.htm and unzip them. One can then use individual programs by:

(1) Click Windows “start” menu;
(2) Click “Run…”;
(3) A dialog will jump out. Type “cmd” in the dialog.
(4) A command window will jump out. Use command “cd” to enter the directory

where CisGenome executables are installed.

(5) Run CisGenome functions by typing their names and required parameters. For example:

```
> genome_getseq –i … -d …
```

Notice that if you don't know how to set parameters for a function, you can type the function name only. A usage message will be displayed for your reference.

1.2.2 Installation on Unix

To install CisGenome on Unix, download the source codes from http://biogibbs.stanford.edu/~jihk/CisGenome/index.htm and unzip them. You need to compile the source codes by running makefile. Before running the makefile, make sure that you have gcc

Run makefile by typing:

```
> makefile
```

You need to have gcc in order to compile the source files. After the programs are installed, one can use the individual programs by typing their names and parameters. For example:

```
> genome_getseq –i … -d …
```

Notice that if you don't know how to set parameters for a function, you can type the function name only. A usage message will be displayed for your reference.

1.3 List of Functions

1.3.1 Genomics Toolbox I – Establishing Local Genome Database

(1) *genome_encode* – Converting genome sequences from FASTA format to CisGenome format (*.sq files). *.sq files are binary files. Each byte in the *.sq file contains two nucleotides. These files are the starting point for most CisGenome sequence manipulation functions.

(2) *genome_markovbg (motifmap_matrixscan_genome_bg)* – Fitting a set of Markov models as the sequence generating model. The models are fitted from *.sq sequence files and have different parameters for different genomic loci. Fitted models can be used in various downstream analyses, e.g., they can be used as background models when mapping transcription factor binding motifs to genomic sequences.

(3) *genome_codephastcons_v2* – Converting PhastCons scores to CisGenome format (*.cs files). *.cs files are binary files. Each byte in the *.cs file contains the conservation score for a single genomic position (i.e., a single base pair). The PhastCons scores are converted linearly from interval [0, 1] to interval [0, 255]. A larger score corresponds to a more conserved status.

(4) *genome_footprint* – Computing a user-specified conservation scores from genome-wide MULTIZ multiple species alignments. Users can specify the meaning of "conservation". For example, "if three contiguous bases are identical in human, mouse and dog, and if the 3-mer has at least two identical matches from the other three species chicken, frog or zebrafish, then the 3-mer is conserved". Conservation scores are stored in CisGenome format (*.cs files). Each byte in the *.cs file corresponds to a single position in the genome. A score "255" corresponds to a conserved status, and "0" corresponds to a non-conserved status.

(5) *genome_conservebg*, *genome_conservecs* – Computing a conservation score for each genomic position based on a sliding window percent identity method. The scores range from 0 to 255, with a larger score corresponding to a more conserved status. Scores are stored in CisGenome format (*.cs files) which are binary files. Each byte in the *.cs file corresponds to a single position in the genome.

(6) *genome_csgetdistn* – Get empirical distribution of conservation scores.

(7) *refgene_encode* – Converting gene annotations from UCSC RefGene format to CisGenome format. The output file is part of the local annotation database.

(8) *refflat_encode* – Converting gene annotations from UCSC RefFlat format to CisGenome format. The output file is part of the local annotation database.

(9) *refgene_pickspeciesspecific* – Selecting annotated transcripts that origin from a specific species.

(10) *genome_codingCDS* – Creating protein coding region indicator files (*.cds). The *.cds files are binary files. Each byte corresponds to a single position in the genome. The indicator = 1 if the position is within a coding region, the indicator = 0 otherwise.

1.3.2 Genomics Toolbox II – Sequence and Annotation Retrieval

(1) *genome_getseq, genome_getseq_c* – Retrieving DNA sequences from specified genomic regions. Sequences are retrieved from local genome databases (i.e., *.sq files created in section 2), and the retrieved sequences will be stored in a FASTA file.

(2) *genome_getseqcs, genome_getseqcs_c* – Retrieving DNA sequences and conservation scores from specified genomic regions. Sequences and conservation scores are retrieved from local genome databases (i.e., *.sq and *.cs files created in section 2). The retrieved DNA sequences will be stored in a single FASTA file. For each sequence, its corresponding conservation score will be stored in a separate file.

(3) *genome_getmaskedseq, genome_getmaskedseq_c* – Retrieving DNA sequences for specified genomic regions. Users can choose to mask protein coding regions or regions with low conservation level by ‘N’.

(4) *genome_getmaskedreg, genome_getmaskedreg_c* – Filter genomic regions by their repeat, protein coding and cross-species conservation properties.

(5) *genome_fastaseqmask* – Masking specific motifs from sequences. Both input and output sequences are stored in FASTA files. Masked nucleotides will be replaced by ‘N’ in the output file.

1.3.3 Genomics Toolbox III – Annotating specified genomic regions

(1) *genome_getcsgcsummary* – Get nucleotide occurrence frequencies and conservation score distribution for specified genomic regions.

(2) *refgene_getnearestgene* – Associating genomic regions with neighboring genes.

(3) *refgene_gettssaround* – Get regions surrounding transcription start sites.

(4) *refgene_getaffy* – Get Affymetrix probeset IDs for a set of genes.

(5) *reflex_getmultiortholog, refgene_getmultiortholog* – Search for orthologs of a set of genes.

1.3.4 Microarray Toolbox – Gene selection

(1) *powexpress* – Selecting genes that show specific expression patterns.

(2) *powexpress_getspecificprobe* – Get raw expression data for specified probesets.

(3) *powexpress_getnrprobe* – Remove redundant probesets from the raw data file.

1.3.5 ChIP-chip Toolbox – ChIP-chip peak detection

(1) *tilemap_importaffy* – import data from Affymetrix arrays.

(2) *tilemap_norm* – quantile normalization.

(3) *tilemap* – detect binding regions

(4) *tilemap_extract* – retrieve probes and summary statistics in user-specified regions. The retrieved data can be easily loaded into R, Matlab etc. for visualization.

1.3.6 De novo motif discovery Toolbox

(1) *flexmodule_motif* – *De novo* motif discovery based on a collapsed Gibbs

Motif Sampler.

(2) *flexmodule_tnum – De novo* motif discovery that favors TFBS physically clustered together.

(3) *flexmodule – De novo* motif/module discovery where users can specify module structures (this function will be discussed elsewhere).

1.3.7 Known Motif Mapping Toolbox

(1) *motifmap_consensusscan_genome*, *motifmap_consensusscan* – Mapping a consensus motif to genomic regions or FASTA sequences.

(2) *motifmap_matrixscan_genome*, *motifmap_matrixscan* – Mapping a position specific weight matrix (PWM) to genomic regions or FASTA sequences.

(3) *motifmap_filter_genome* – Filtering TFBS by conservation and protein coding characteristics.

(4) *motifmap_getsitearound* – Extending TFBS to include flanking regions.

(5) *motifmap_getsitearoundcs* – Getting average conservation scores for positions within and around TFBS.

(6) *motifmap_matrixscan_summary* – Computing relative enrichment level for a list of PWMs in target regions as compared to control regions.

(7) *motifmap_matrixscan_enrich* – Computing relative enrichment level of a PWM in ranked and tiered regions.

1.3.8 Cross-species comparison toolbox

(1) *malign_genome_prepareortholog* – Prepare the extended window surrounding ortholog genes.

(2) *malign_genome_blasthit* – Generate cross-species alignments.

(3) *malign_motifmap* – Mapping transcription factor binding motifs to alignments.

(4) *malign_modulemap* – Selecting modules that contain specific TFBS.

1.4 A Quick Start – Analysis of a ChIP-chip Experiment

In this section, we use an example to show how CisGenome can be used to support a complete ChIP-chip study.

1.4.1 Microarray gene selection.

We first use powexpress to select SHH responsive genes.

```
>powexpress -d shhdata1.txt -a 3mousechipsinfo.txt -c shhcompinfo1.txt -o shh_8som_pos
```

1.4.2 ChIP-chip peak detection.

Using the selected genes, we designed a customer tiling array and performed ChIP-chip analysis on the array. The goal of the analysis is to identify GLI binding targets. TileMap was applied to detect binding regions.

```
>tilemap tilemap_arg.txt
```

1.4.3 Get sequence

After peak detection, we retrieved sequences in the binding regions.

```
>genome_getseq_c -d /data/genomes/mouse/mm6/ -s mouse -i Gli_cut3-cut4_e200_cod.txt -o Gli_cut3-cut4_e200.fa
```

1.4.4 De novo motif discovery

We then performed de novo motif discovery on high quality regions.

```
> flexmodule_motif flexmodule_allm10l9_arg.txt
```

1.4.5 Ascertain the key binding motif

From the discovered motifs, the key binding motif can be ascertained by checking motifs' relative enrichment levels.

```
> motifmap_matrixscan_genome_summary -mr motiflist.txt -gd /data/genomes/mouse/mm6/ -i Gli_cut3-cut4_e200_cod.txt -n Gli_matchneg.cod -o motifs_enrich_match.txt -b 3 -bt genome -bd /data/genomes/mouse/mm6/markovbg/S100000_W1000000 -bs 100000 -c 40 -cd /data/genomes/mouse/mm6/conservation/genomelab/cs/
```

1.4.6 Refining peak cutoff

After the key motif is found, it can be used to refine the peak cutoff.

```
> motifmap_matrixscan_genome_enrich -m Gli_matrix.txt -gd /data/genomes/mouse/mm6/ -i Gli_cut3_e200_cod.txt -n Gli_matchneg.cod -s 10 -o Gli_tierenrich.txt -r 500 -b 3 -bt genome -bd /data/genomes/mouse/mm6/markovbg/S100000_W1000000 -bs 100000 -c 40 -cd /data/genomes/mouse/mm6/conservation/genomelab/cs/
```

1.4.7 Check GC content and conservation

Based on the refined cutoff, 30 regions were defined as final GLI binding regions. Next, we check GC content and conservation scores of the regions.

```
> genome_getcsgcsummary -gd /data/genomes/mouse/mm6/ -i Gli_cut3_e200_top30_cod.txt –o Gli_mm6_summary -c -1 -cd /data/genomes/mouse/mm6/conservation/genomelab/cs/
```

1.4.8 Annotate bining regins using their nearst gene.

```
> refgene_getnearestgene -d /data/genomes/mouse/mm6/annotation/refFlat_sorted.txt -dt 1 -s mouse -i Gli_cut3_e200_top30_cod.txt -o Gli_top30_gene.txt -r 0 -up 100000 -down 100000
```

1.4.9 Find ortholog genes

```
> getrefgenemultiortholog -i Gli_top30_gene.txt -c 6 -d orthologsetting3_mouse.txt -o Gli_top30_3way
```

1.4.10 Generate cross-species alignments and annotate them by TFBS

```
> malign_genome_prepareortholog -i Gli_top30_3way.nsomap -o Gli_top30_3way_foraln.txt -n 4 -sf 1 -r 0 -up 50000 -down 50000
```

```
> malign_genome_blasthit malign_genome_arg_u.txt
```

```
> malign_motifmap malign_motifmap_arg_u.txt
```

1.4.11 Map Gli matrix to the whole mouse genome to make predictions

```
> motifmap_matrixscan_genome -m Gli_matrix.txt –gd /data/genomes/mouse/mm6/ -i mm6_cod.txt -o Gli_mm6.map -r 500 -b 3 -bt genome -bd /data/genomes/mouse/mm6/markovbg/S100000_W1000000 -bs 100000 -c 40 -cd /data/genomes/mouse/mm6/conservation/genomelab/cs/
```

1.4.12 Search for potential cofactors

```
> motifmap_matrixscan_genome_summary -mr transfac_motiflist.txt -gd /data/genomes/mouse/mm6/ -i Gli_cut3_e200_top30_cod.txt -n Gli_matchneg.cod -o motifs_enrich_match.txt -b 3 -bt genome -bd /data/genomes/mouse/mm6/markovbg/S100000_W1000000 -bs 100000 -c 40 -cd /data/genomes/mouse/mm6/conservation/genomelab/cs/
```

2. GENOMICS TOOLBOX I – ESTABLISHING LOCAL GENOME DATABASES

2.1 Introduction to Local Genome Database

Before one can use most sequence manipulation functions of CisGenome, one needs to have genomes and annotations stored locally and stored in CisGenome format. The locally stored genomes and annotations are referred to as *local genome database* thereafter. The typical structure of a local genome database is shown in the next page.

A local genome database is only required when one wants to do sequence manipulations. The use of microarray toolbox (e.g., powexpress), ChIP-chip toolbox (e.g., tilemap) does not require the availability of local genome database. If one has sequences in FASTA format, some sequence analysis functions (e.g., motif mapping, de novo motif discovery) can be applied without the local genome database. When one wants to get sequences for specified genomic regions, gene annotations, and wishes to do genome-scale transcription factor binding motif mapping, however, one needs to establish the local genome database first.

The easiest way to establish a local genome database is to download it from CisGenome website, where sequences and annotations are provided for several commonly used genomes (including human and mouse) in CisGenome format. Once downloaded and uncompressed, the database should be ready for use.

Not all genomes, however, have a database available on CisGenome website. When users want to use these genomes, or when users want to update their genome databases by themselves before the update is released by us, users can use CisGenome functions to establish their own databases. The creation of a local genome database is a one time deal. Once it is established, one can use it in all subsequent sequence manipulations and in different research projects. Although creating such a database from scratch requires some work and the database is not required by all CisGenome functions, we strongly recommend users to establish it when they start to use CisGenome, since the database will make many downstream sequence analyses much more convenient, flexible and efficient. Starting from section 2.3, we will introduce how such a database can be established by yourself.

CisGenome Local Database Structure

```
– genomes                               /* root */
   – mouse                              /* mouse database */
      + mm7                             /* database for mm7 assembly */
      + mm6                             /* database for mm6 assembly */
   + dog                                /* dog database */
   – human                              /* human database */
      + hg18                            /* database for hg18 assembly */
      – hg17                            /* database for hg17 assembly */
         –> chrlist.txt                 /* list of chromosomes */
         –> chrlen.txt                  /* chromosome lengths */
         –> chr1.sq                     /* sequence for chromosome 1 */
         –> …
         – annotation                   /* gene annotations */
            -> refGene_sorted.txt       /* human refGene annotation */
            -> refFlat_sorted.txt       /* human refFlat annotation */
            -> xenoRefGene_sorted.txt
                /* refGene annotation using RNAs from other species */
            -> xenoRefFlat_sorted.txt
                /* refFlat annotation using RNAs from other species */
            –> mouseRefGene_sorted.txt
                /* refGene annotation using RNAs from mouse */
            –> …
         – alignment                    /* crossspecies alignments */
            –> chr1.maf                 /* chromosome1 alignments */
            –> …
         – cds                   /* protein coding region indicators */
            –> chr1.cds          /* CDS indicators for chromosome 1*/
            –> …
         – conservation          /* cross-species conservation scores */
            – phastcons          /* PhastCons scores */
               –> chr1.cs        /* PhastCons scores for chromosome 1*/
               –> …
            – footprint          /* User-computed footprint scores  */
               –> chr1.cs        /* footprint scores for chromosome 1*/
               –> …
            – genomelab          /* genomelab conservation scores */
               – bg              /* background models for computing
                                    genomelab conservation scores */
                  –> chr1.bg
                  –> chr1.cov
                  –> …
               – cs               /* scores */
                  –> chr1.cs   /* scores for chromosome 1*/
                  –> …
         – markovbg              /* Markov background sequence model */
```

```
                    – S100000_W1000000 /* background model computed using a 1Mb
                          sliding window and the sliding step size is 100kb */
                        + 0             /* 0-order Markov models */
                        – 3             /* 3-order Markov models */
                            – chr1      /* 3-order Markov model for chr1*/
                                –> 1_f.txt   /* forward Markov model for bin 1*/
                                –> 1_b.txt   /* backward Markov model for bin 1*/
                                –> …
                            – chr2      /* 3-order Markov model for chr2*/
                            – ………
```
Notes:
+ : a closed directory.
– : an open directory for which all its contents are shown.
–>: a file

For each species and each assembly, one can establish a database following the structure shown above. Taking human hg17 (May 2004 NCBI assembly build 35) as an example, in the directory hg17 we have the following information stored.

(1) *chrlist.txt* stores a list of chromosome names in a text file, in the following format:
chr1
chr2
…
chrX
chrY

(2) *chrlen.txt* stores the chromosome length for each chromosome, in the following format:
245522847
243018229
…
154824264
57701691

There is a one-to-one correspondence between lines in chrlist.txt and lines in chrlen.txt, e.g., the length of chrX is 154824264 base pairs.

(3) *chr*.sq* files (required) store genome sequences. A *.sq file is a binary file. Each byte contains information for two nucleotides. Nucleotides are coded as below:
```
A: 0000
C: 0001
G: 0010
T: 0011
a: 0100
c: 0101
```

```
g: 0110
t: 0111
N: 1000
```

Here, ‘a’, ‘c’, ‘g’, ‘t’ represent soft-masked repeat sequences. Due to the coding of sequences, the human genome (~3Gbp) will require about 1.5GB space to be stored in a hard disk. The length of each *.sq file is approx. half of the length of corresponding chromosome, e.g., the size of chrX.sq is 77,412,132 bytes.

(4) *annotation* is a subdirectory that stores gene structure information, including *refGene_sorted.txt*, *refFlat_sorted.txt*, *xenoRefGene_sorted.txt*, *xenoRefFlat_sorted.txt*, *mouseRefGene_sorted.txt*, etc. These files have almost the same format as UCSC files *refGene.txt*, *refFlat.txt* etc. (please refer to http://genome.ucsc.edu/goldenPath/gbdDescriptions.html)

The differences between **_sorted.txt* and **.txt* include: (i) in **_sorted.txt*, annotations are sorted according to their chromosomal location; (ii) in **_sorted.txt*, chromosome names are replaced by a numerical ID, e.g., chr1 replaced by 1, and chrX replaced by 23. Usually, the numerical ID is the same as the line # in the chrlist.txt file.

Among various annotations, *refGene_sorted.txt* and *refFlat_sorted.txt* are transcripts that origin from the species in question, (e.g., *refGene_sorted.txt* in human/hg17/ contains gene structures for human transcripts; *refGene_sorted.txt* in mouse/mm6/ contains gene structures for mouse transcripts, etc.). *xenoRefGene_sorted.txt* and *xenoRefFlat_sorted.txt* contain transcripts that come from all other species. *mouseRefGene_sorted.txt* contains transcripts from mouse only, etc.

Users can add their own gene annotations, as long as the annotations have the same format as UCSC *refGene.txt* or *refFlat.txt.*

(5) *alignment* (optional) is a subdirectory that stores cross-species alignments in *.maf format (http://genome.ucsc.edu/goldenPath/help/maf.html). Usually, these alignments can be downloaded from UCSC Genome browser (e.g., MULTIZ Multiple alignments of 16 vertebrate genomes with Human).

(6) *conservation* (optional) is a subdirectory that stores cross-species conservation scores. The scores can be computed using different ways. For example, subdirectory *phastcons* contains phastCons scores; subdirectory *footprint* contains customized conservation scores computed by CisGenome, etc.

In each subdirectory, the conservation scores are stored in *.cs files. A *.cs file is a binary file. Each byte in the file corresponds to a single position in the genome. Therefore, the size of chr1.cs is the same as the length of chromosome 1, and one needs approx. 3GB space to store conservation scores for human genome, etc. Each byte contains a score from 0 to 255, with 255 corresponding to the most conserved status. Usually, the bigger the score, the more conserved a position is.

(7) *cds* (optional) is a subdirectory that stores protein coding region indicators. The indicators are stored in *.cds files. A *.cds file is a binary file. Each byte in the file corresponds to a single position in the genome. If the position is located in a protein coding region, the byte is equal to 1, otherwise the byte is equal to 0.

<u>*Hint*</u>: Users may have noticed that *conservation* and *cds* scores have similar storage structures. This is designed to facilitate downstream analysis so that different kinds of information can be processed in a common way. Indeed, users can create their own scores (such as CpG island indictors, histone modification scores, etc.) and store them in the same format as the scores here. User defined scores can then be used in the same way as *conservation* and *cds* scores here.

(8) *markovbg* (optional) is a subdirectory that stores background Markov models fitted to describe the sequence generating process. *S100000_W1000000* contains models that are fitted using a 1Mb window sliding along the chromosomes at a 100000bp step size. *S100000_W1000000/0* contains 0 order Markov models, and *S100000_W1000000/3* contains 3rd order Markov models. [n]_f.txt is the forward model for the nth sliding window, and [n]_b.txt is the backward model for the nth sliding window. In a forward model, say for sequences AGCTGA, the transition probabilities are fitted by counting AGC->T, GCT->G, CTG->A, etc. In a backward model, the transition probabilities are fitted by counting GCT->A, CTG->G, TGA->C etc. Each kth order model is represented by a 4^kx4 matrix. The four columns correspond to A, C, G, T respectively, and the rows are indexed by the following word index:
A = 0;
C = 1;
G = 2;
T = 3;
ACT = $0*4^2+1*4^1+3*4^0$
etc.

2.2 List of Functions

(1) *genome_encode* – Converting genome sequences from FASTA format to CisGenome format (*.sq files). *.sq files are binary files. Each byte in the *.sq file contains two nucleotides. These files are the starting point for most CisGenome sequence manipulation functions.

(2) *genome_markovbg (motifmap_matrixscan_genome_bg)* – Fitting a set of Markov models as the sequence generating model. The models are fitted from *.sq sequence files and have different parameters for different genomic loci. Fitted models can be used in various downstream analyses, e.g., they can be used as background models when mapping transcription factor binding motifs to genomic sequences.

(3) *genome_codephastcons_v2* – Converting PhastCons scores to CisGenome format (*.cs files). *.cs files are binary files. Each byte in the *.cs file contains the conservation score for a single genomic position (i.e., a single base pair). The PhastCons scores are converted linearly from interval [0, 1] to interval [0, 255]. A larger score corresponds to a more conserved status.

(4) *genome_footprint* – Computing a user-specified conservation scores from genome-wide MULTIZ multiple species alignments. Users can specify the meaning of “conservation”. For example, “if three contiguous bases are identical in human, mouse and dog, and if the 3-mer has at least two identical matches from the other three species chicken, frog or zebrafish, then the 3-mer is conserved”. Conservation scores are stored in CisGenome format (*.cs files). Each byte in the *.cs file corresponds to a single position in the genome. A score “255” corresponds to a conserved status, and “0” corresponds to a non-conserved status.

(5) *genome_conservebg, genome_conservecs* – Computing a conservation score for each genomic position based on a sliding window percent identity method. The scores range from 0 to 255, with a larger score corresponding to a more conserved status. Scores are stored in CisGenome format (*.cs files) which are binary files. Each byte in the *.cs file corresponds to a single position in the genome.

(6) *genome_csgetdistn* – Get empirical distribution of conservation scores.

(7) *refgene_encode* – Converting gene annotations from UCSC RefGene format to CisGenome format. The output file is part of the local annotation database.

(8) *refflat_encode* – Converting gene annotations from UCSC RefFlat format to CisGenome format. The output file is part of the local annotation database.

(9) *refgene_pickspeciesspecific* – Selecting annotated transcripts that origin from a specific species.

(10) *genome_codingCDS* – Creating protein coding region indicator files (*.cds). The *.cds files are binary files. Each byte corresponds to a single position in the genome. The indicator = 1 if the position is within a coding region, the indicator = 0 otherwise.

2.3 Establishing Database Step I – Download Sequences and Annotation

The first step to establish a local genome database is to download sequences and gene annotations. Both can be downloaded from UCSC Genome browser (http://hgdownload.cse.ucsc.edu/downloads.html). For example, to establish human hg17 local database, one can download chromFa.zip from UCSC to {Local Database Path}/genomes/human/hg17/, and download refFlat.txt.gz, refGene.txt.gz, xenoRefFlat.txt.gz, xenoRefGene.txt.gz, etc. to {Local Database Path}/genomes/human/hg17/annotation/. If needed, one may also download phastCons conservation scores to {Local Database Path}/genomes/human/hg17/conservation/phastCons, and MULTIZ alignments to {Local Database Path}/genomes/human/hg17/alignment.

When downloading sequences, we recommend the use of soft-masked sequences since it keeps the full sequence information. In CisGenome, the differences between capital letters ‘A’, ‘C’, ‘G’, ‘T’ and little ones ‘a’, ‘c’, ‘g’, ‘t’ can be handled by *.sq files (section 2.3).

After downloading necessary files, uncompress them for later use.

2.4 Establishing Database Step II – Coding Genome Sequences

The goal of this step is to convert genomes sequences from FASTA (*.fa, ASCII) files to *.sq files (binary). *.sq files are the basis of CisGenome sequence manipulations. The conversion can be achieved through following steps.

(1) Create a *chrlist.txt* file that list chromosome names in an increasing order as follows:

```
chr1
chr2
…
chrX
chrY
```

For human hg17, save this file to {Local Database Path}/genomes/human/hg17/; for mouse mm6, save this file to {Local Database Path}/genomes/mouse/mm6/, etc. Make sure that for each chromosome (say chr[#]) listed in the file, the corresponding chr[#].fa file can be found in the same directory. Currently, chr[#]_random, chrUn and chrM are not supported by CisGenome. Please do not include these chromosomes into the list.

(2) Run *genome_encode* as follows.

genome_encode –d [Path where *.fa are stored] –o [Path where *.sq will be stored]

For example

```
>genome_encode –d /data/genomes/human/hg17/ -o /data/genomes/human/hg17/
```

(3) After *genome_encode* has been run, one should be able to find a set of *.sq files and a file named *chrlen.txt* in the output path (i.e., the directory specified by –o option). The *chrlen.txt* is generated by the program automatically and contains chromosome lengths for all chromosomes listed in the *chrlist.txt*. The first line in *chrlen.txt* is the length for the first chromosome listed in the *chrlist.txt*, the second line in *chrlen.txt* is the length for the second chromosome in *chrlist.txt*, etc.

(4) The *.fa files will not be used anymore by CisGenome, and users can remove them from the hard disk to save storage space.

2.5 Establishing Database Step III – Coding Gene Annotations

The goal of this step is to create gene annotations for CisGenome to use. The annotations can be used to in various downstream analyses such as annotating ChIP-binding regions, extracting sequences around transcription start sites, etc. Again, taking human hg17 as an example, users can create the annotations as follows.

(1) Make sure that annotation files *refGene.txt*, *refFlat.txt*, *xenoRefGene.txt*, *xenoRefFlat.txt* downloaded from UCSC Genome Browser are stored in {Local Database Path}/genomes/human/hg17/annotation/.

(2) For files in *refGene* format, including *refGene.txt* and *xenoRefGene.txt*, run *refgene_encode* as follows:
refgene_encode -d [Path of the UCSC annotation file] -o [Path of the output file] -s [Species name] -n [Number of chromosomes for the given species]

For example:
```
>refgene_encode -d /data/genomes/human/hg17/annotation/refGene.txt -o /data/genomes/human/hg17/annotation/refGene_sorted.txt -s human -n 24
```

Note that species name can be one of {human, mouse, dog, cow, chicken, zebrafish}. We will keep adding other species to our support list. The number of chromosomes is 24 for human (chr1~chr22, chrX, chrY), 21 for mouse (chr1~chr19, chrX, chrY), etc.

(3) For files in *refFlat* format, including *refFlat.txt* and *xenoRefFlat.txt*, run *refflat_encode* as follows:
refflat_encode -d [Path of the UCSC annotation file] -o [Path of the output file] -s [Species name] -n [Number of chromosomes for the given species]

For example:
```
>refflat_encode -d /data/genomes/human/hg17/annotation/refFlat.txt -o /data/genomes/human/hg17/annotation/refFlat_sorted.txt -s human -n 24
```

(4) (Optional, supported in UNIX only, and only refGene format is supported).
Some times it is useful to group gene annotations according to their species origins. One can run *refgene_pickspeciesspecific* to do the job.
refgene_pickspeciesspecific -i [the file that contains species-specific Refseq IDs] -d [the file that contains xenoRefGene_sorted annotations] -o [output file]

For example, if we want to pick up mouse refGenes that are aligned to human genome hg17 assembly, we can use
```
>refgene_pickspeciesspecific -i /data/genomes/mouse/mm6/annotation/refGene_sorted.txt -d /data/genomes/human/hg17/annotation/xenoRefGene_sorted.txt -o /data/genomes/human/hg17/annotation/mouseRefGene_sorted.txt
```

As a result, all transcripts that align to human genome and that origin from mouse will be selected from human *xenoRefGene_sorted.txt*, and the selected transcripts will be stored in *mouseRefGene_sorted.txt*, and these annotations are based on human coordinates.

(5) After running above steps, check to make sure that appropriate *_sorted.txt files such as *refGene_sorted.txt*, *refFlat_sorted.txt*, *xenoRefGene_soreted.txt*, *xenoRefFlat_soretd.txt* are generated.

Hint: Users may use their own gene structure annotations instead of annotations provided by *refGene.txt* or *refFlat.txt*. To use their own annotations, users only need to prepare their annotation files in the UCSC *refGene* or *refFlat* format, they can then run *refgene_encode* or *refflat_encode* as above. As a simple example, one can replace *refGene.txt* by *ensGene.txt* (Ensembl gene annotations) downloaded from UCSC Genome browser.

2.6 Establishing Database Step IV – Creating Markov Background

The goal of this step is to fit Markov models that can be used to describe sequence generating processes. The fitted models can be used as background models for detecting transcription factor binding sites (refer to section 8). Models can be fitted as below.

(1) Make sure to create a directory to store the fitted models. For example, one can create a directory /data/genomes/human/hg17/markovbg/S100000_W1000000/3/ to store the 3rd order Markov models fitted using parameters S=100000 and W=1000000 (which will be discussed below).

Make sure that *chrlist.txt* and *chrlen.txt* are available in the directory where sequences (*.sq) files are stored.

(2) Run *motifmap_matrixscan_genomebg* as follows:

motifmap_matrixscan_genomebg -d [Path where *.sq sequence files are stored] -o [Path where fitted model will be stored] -b [The order of Markov models] -s [Step size] -w [Window size]

For example:

```
>motifmap_matrixscan_genomebg -d /data/genomes/human/hg17/ -o /data/genomes/human/hg17/markovbg/S100000_W1000000/3/ -b 3 -s 100000 -w 1000000
```

What this function does is to form a window that is W (here W=1000000) base pair long. The window will be sliding along the genome using a step size S (here S=100000 bp). All the non-repeat nucleotides in the window will then be used to fit a forward Markov model and a backward Markov model, both of order B (B=3 here). For sequences 5'-abcd-3', a 3rd order forward model describes transition probabilities abc->d, and a 3rd order backward model describes transition probabilities bcd->a. The models are fitted using sequences stored in the sequence directory which is specified by "–d" option. After running the program, a set of subdirectories (chr1, chr2, …, chrX, chrY) will be created in the output path which is specified by "–o" option. Files in the subdirectory "chr[#]" are models fitted for chromosome [#]. For example, /data/genomes/human/hg17/markovbg/S100000_W1000000/3/chr[#]/[n]_f.txt is the forward model for the nth sliding window in chr[#], and /data/genomes/human/hg17/markovbg/S100000_W1000000/3/chr[#]/[n]_b.txt is the nth backward model. Both [n]_f.txt and [n]_b.txt are centered at the interval [(n-1)*100000+1, n*100000] which is extended equally to both ends to cover a 1Mb region. Coordinates beyond the range of chromosomes are discarded.

Each [n]_f.txt or [b]_f.txt file contains a 4^Bx4 matrix. The number in the *i*th row and *j*th column is the transition probability from *i* to *j*. The four columns in the matrix correspond to *j*=A, C, G, T respectively, and the rows are indexed by the following word index *i*:

(A = 0; C = 1; G = 2; T = 3)

GTC = $2*4^2+3*4^1+1*4^0$
ACT = $0*4^2+1*4^1+3*4^0$
etc.

2.7 Establishing Database Step V – Coding Conservation Scores

The goal of this step is to create cross-species conservation scores in CisGenome format. The conservation scores can be used in downstream analysis such as finding conserved transcription factor binding sites, selecting conserved sequences for de novo motif discovery, etc. Users can both use phastCons scores downloaded from UCSC Genome browser and compute their own scores based on their specific needs.

In order to use these scores in downstream analysis, all scores need to be stored in *.cs format (a *.cs file is a binary file, one byte -> one base pair, score ranges from 0 to 255, the larger the more conserved; refer to section 2.2 for details).

2.7.1 Coding PhastCons Scores

One can convert phastCons scores to *.cs format using the following steps.

(1) Make sure that phastCons scores are downloaded from UCSC Genome browser and stored in an appropriate place. For example, for human hg17, one can store them in /data/genomes/human/hg17/conservation/phastcons/.

Make sure that *chrlist.txt* and *chrlen.txt* are available in the directory where genome sequences (*.sq files) are stored.

Make sure that for each chromosome listed in the *chrlist.txt*, say chr[#], there is a raw phastCons conservation score file named *chr[#]*.

(2) Run *genome_codephastcons_v2* as follows:

genome_codephastcons_v2 –d [Path where genome sequences *.sq are stored] –c [Path where downloaded phastCons scores are stored] –o [Path where the converted scores *.cs will be stored]

For example:

```
>genome_codephastcons_v2 -d /data/genomes/human/hg17/ -c /data/genomes/human/hg17/conservation/phastcons/ -o /data/genomes/human/hg17/conservation/phastcons/
```

(3) After running *genome_codephastcons_v2*, there should be a collection of *.cs files generated in the output directory which is specified by "-o" option. At this stage, the raw phastCons score files downloaded from UCSC Genome browser will not be used by CisGenome anymore. To save storage space, users can remove them from the disk.

(4) To facilitate the future use of conservation scores, run *genome_csgetdistn* to get the empirical distribution of conservation scores:

genome_csgetdistn -d [Path where the conservation scores *.cs are stored] -s [Species name] -l [File that contains chromosome lengths] -i [File that list all chromosomes] -o [output file]

For example:

```
>genome_csgetdistn -d /data/genomes/human/hg15/conservation/cs/ -s human -l /data/genomes/human/hg15/chrlen.txt -i /data/genomes/human/hg15/chrlist.txt -o
```

csstat.txt

What genome_csgetdistn returns is a text file (in this example, csstat.txt) that contains 256 lines. The line n counts the total number of base pairs for which conservation score = n-1. Using the file, one can easily determine a cutoff c so that, say, the total number of positions where the score >= c accounts for 10% of the genome. The cutoff can then be used in various applications to define “conserved sequences”.

2.7.2 Creating Customer FootPrint Scores

Users can generate their customized conservation scores from multiple species alignments. For example, to study transcription factor binding sites with deep conservation, one can define a conservation score as follows. The score=255 if “a site contains at least 6 contiguous base pairs that are identical in mouse, human, dog, cow, chicken, frog and zebrafish”; otherwise, the score=0.

Such customized conservation scores will be called “Footprint” scores thereafter. Given multiple species alignments, Footprint scores are defined by the following parameters: (i) a window size S, (ii) a reference species; and (iii) a set of conservation criteria. As an example, if one defines S = 6, reference species = mouse, and conservation criteria as “human>=1, dog+cow>=1, and chicken+frog+fish>=2”, then the Footprint score is computed as follows. A 6bp window will be sliding along the alignments. For each position, the 6-mer from all species will be compared with the 6-mer from the reference species (here mouse). If the 6-mer from human is identical to the 6-mer in mouse (i.e., human>=1), at least one of the 6-mers from dog and cow is identical to the mouse 6-mer (i.e., dog+cow>=1), and at least two of the 6-mers from chicken, frog and fish are identical to the mouse 6-mer (i.e., chicken+frog+fish>=2), then the 6 nucleotides in the window are conserved. A conservation score = 255 will be attached to each of the 6 bases. If the conservation condition is not satisfied, then the window is not a conserved window, and nucleotides within the window will be assigned a conservation score = 0 except for those bases which are covered by other conserved windows. The figure below illustrates conserved scores computed in this way (“*”= 255, “0”= 0).

```
Score   000000*******000000000000********0000000000000000000000000*********
MOUSE   TTGTTACAAAACAAAGTGGGAGGTACCACCCAGACACATATGTTGATTCGCAGGAGGCTGTTTACAT
HUMAN   TTG-TACAAAACATAGTGAAAGA-ACCACCCAGTC-AATTTGTCGATTCGTAGGAG-CTGTTTACAT
DOG     TTGTTTCAAAACATAGTGCGAGATACCACCCAGAG-AATCTGTCGATTCGT-GGAG-CTGTTTACAT
COW     TTGTTTCAAAACAAAGTGAGACA-ACCACCCAGAC-CATGTGTCGATTCGTAGGAG-CTCTTTACAT
CHICKEN TTGTTACAAAACAAAGT--GAGT-ACCACCCAGAG-AATATGTTGAATCGT-GAAG-CTGTTTACAT
FROG    TTG-TACAAAACATACT--TCGA-ACCACCCAGAG-CATCTGTTGATCCGT-GAAG-----------
FISH    ------------------------GCCACCCAGCC-AGTATGTCGATTTAC-GAGG-TTGTTTACAT
```

In addition to S, reference species and conservation criteria, there is another parameter “target species” that specifies for which species the conservation score is computed. If target species = mouse, then the computed conservation scores will be attached to nucleotides in mouse genome. A collection of *.cs files will be generated, and their combined size will be equal to the size of mouse genome. In other words, each position in mouse genome will be assigned a footprint score derived from above procedures. On the other hand, if target species = human, then the computed conservation scores will be linked to human genome. A collection of *.cs files will be generated, and their combined size will be equal to the size of human genome. Each position in human genome will be assigned a footprint score derived as above, although the score was computed using mouse as the reference species (i.e., there is no requirement that reference species = target species).

Users can follow the steps below to generate Footprint scores.

(1) Make sure that multiple species alignments in MAF format are downloaded from UCSC Genome browser and stored in an appropriate place. For example, for human hg17, one can store them in /data/genomes/human/hg17/alignment/.

Make sure that *chrlen.txt* for both reference species and target species are available.

(2) Prepare a tab-delimited parameter file, say genome_footprint_arg.txt, as below:

```
Alignment_Dir /data/genomes/human/hg17/alignment/

Species_Num   8
ID  0   hg
ID  1   panTro
ID  2   mm
ID  3   rn
ID  4   canFam
ID  5   galGal
ID  6   fr
ID  7   danRer

Reference  0
Ref_Species   human
Ref_Chr_Size  /data/genomes/human/hg17/chrlen.txt

Target 0
Tar_Species   human
Tar_Chr_Size  /data/genomes/human/hg17/chrlen.txt

Window_Size   3
Comparisons_Num   5
COMP   0>=1
COMP   1>=1
COMP   2+3+4>=2
COMP   5+6+7>=2
COMP   0+1+2+3+4+5+6+7>=7

Output_Path   /data/genomes/human/hg17/conservation/footprint/

Alignment_Suf .maf
Alignment_File    chr1
Alignment_File    chr2
Alignment_File    chr3
Alignment_File    chr4
Alignment_File    chr5
Alignment_File    chr6
Alignment_File    chr7
Alignment_File    chr8
Alignment_File    chr9
Alignment_File    chr10
Alignment_File    chr11
Alignment_File    chr12
Alignment_File    chr13
Alignment_File    chr14
Alignment_File    chr15
Alignment_File    chr16
Alignment_File    chr17
Alignment_File    chr18
Alignment_File    chr19
Alignment_File    chr20
Alignment_File    chr21
Alignment_File    chr22
Alignment_File    chrX
Alignment_File    chrY
```

This file specifies parameters used for computing the footprint score. These parameters include:
(a) Alignment_Dir – Path where MAF alignments are stored.
(b) Species_Num – Number of species in the alignments. For example, if the alignments are MULTIZ alignments for 8 vertebrate species, then the species number is 8.
(c) ID – Numerical ID and string tag for each species. If species number = 8, there should be 8 “ID” lines following the “Species_Num” line. Each ID line has the following format:

```
ID[tab]numerical_id[tab]string_tag
```

String_tag is a tag that is used by the alignments to label species, e.g., hg, mm, canFam, galGal, etc. Numerical_id is a unique number attached to each string_tag. The numerical ID is defined by users. If there are n species, these ids can be selected from 0, 1, 2, …, n-1.

(d) Reference – Numerical ID of the reference species.
(e) Ref_Species – Name of the reference species, e.g., human, mouse, dog, chicken, etc.
(f) Ref_Chr_Size – File that contains the chromosome lengths for reference species.
(g) Target – Numerical ID of the target species.
(h) Tar_Species – Name of the target species, e.g., human, mouse, dog, chicken, etc.
(i) Tar_Chr_Size – File that contains the chromosome lengths for target species.
(j) Window_Size – Window size S.
(k) Comparisons_Num – Number of conservation criteria. If Comparisons_Num=m, the line should be followed by m "COMP" lines.
(l) COMP – Conservation criteria. Each COMP line has the following format:

```
COMP[tab]criteria
```

The criteria should be written in a format such as "2+3+4>=2". Here "2", "3", and "4" are species numerical IDs which are specified in lines starting with "ID". If "2" is linked to mm, "3" is linked to rn and "4" is linked to canFam, then "2+3+4>=2" means that, out of the three species here, at least two of them should have identical bases as the reference species in the sliding window. Otherwise the window will not be defined as conserved.

There is a "AND" relationship between COMP lines. In other words, only windows for which all criteria specified in all COMP lines are satisfied are defined as conserved.

(m) Output_Path – Directory where computed footprint scores will be exported.
(n) Alignment_Suf – Suffix of the alignment files. If the alignment files used to generate footprint scores are ended with *.maf, then Alignment_Suf=.maf
(o) Alignment_File – This species the alignment file that needs to be processed. If there are multiple alignment files, each file should be specified in a separate line starting with "Alignment_File". Each line should have the following format:

```
Alignment_File[tab]chr#
```

The program will automatically go to the directory specified by "Alignment_Dir", find the file named "chr#[Alignment_Suf]", and process it. For example, if

```
Alignment_File chr1
```

then chr1.maf will be processed.

(3) Run *genome_footprint* as below:
genome_footprint [parameter file]

For example:
> genome_footprint genome_footprint_arg.txt

(4) After running *genome_footprint*, one should be able to find a set of *.cs files in the output directory which is specified as "Output_Path" in the parameter file. The size of each *.cs file (i.e., chr1.cs, chr2.cs, etc.) should be equal to the corresponding chromosome length of the target species.

2.7.3 Creating GenomeLab Conservation Scores

Besides phastCons scores and Footprint scores, there is a third option in CisGenome to define conservation scores. This extra option, called "Genomelab conservation scores", is based on a windows percent identity measure. It can complement phastCons scores if the latter are not readily available (e.g., in one of our studies, the phastCons score downloaded from UCSC was missing for a large chunk of chromosome 13. The region contains important genes such as Ptch1 that we are interested in. Since the region did contain high quality cross-species alignments, the phastCons score provided by UCSC was likely to be mistakenly missing for unknown reasons. In such a situation, the tool here allows user to quickly compute their own conservation scores).

Genomelab conservation scores can be computed using following steps.

(1) Make sure that multiple species alignments in MAF format are downloaded from UCSC Genome browser and stored in an appropriate place. For example, for human hg17, one can store them in /data/genomes/human/hg17/alignment/.

Make sure that *chrlen.txt* for both reference species and target species are available.

Make sure that directories that will be used to store the scores are available.

(2) Prepare a parameter file, say genome_conservebg_arg.txt for background computation. The parameter file is tab-delimited and has the following format:

```
Alignment_Dir /data/genomes/human/hg17/alignment/

Species_Num   8
ID  0   hg
ID  1   panTro
ID  2   mm
ID  3   rn
ID  4   canFam
ID  5   galGal
ID  6   fr
ID  7   danRer

Reference  0
Ref_Species    human
Ref_Chr_Size  /data/genomes/human/hg17/chrlen.txt

Comparisons_Num 4
COMP    0   2
COMP    0   4
COMP    0   5
COMP    0   7

BG_Window_Size    1000000
BG_Sliding_Step   10000

Output_Path    /data/genomes/human/hg17/conservation/genomelab/bg/

Alignment_Suf .maf
Alignment_File    chr1
Alignment_File    chr2
Alignment_File    chr3
Alignment_File    chr4
Alignment_File    chr5
Alignment_File    chr6
Alignment_File    chr7
Alignment_File    chr8
Alignment_File    chr9
Alignment_File    chr10
Alignment_File    chr11
Alignment_File    chr12
Alignment_File    chr13
Alignment_File    chr14
Alignment_File    chr15
Alignment_File    chr16
Alignment_File    chr17
Alignment_File    chr18
Alignment_File    chr19
Alignment_File    chr20
Alignment_File    chr21
Alignment_File    chr22
Alignment_File    chrX
Alignment_File    chrY
```

Below are meanings for the parameters.
(a) Alignment_Dir – Path where MAF alignments are stored.
(b) Species_Num – Number of species in the alignments. For example, if the alignments are MULTIZ alignments for 8 vertebrate species, then the species number is 8.
(c) ID – Numerical ID and string tag for each species. If species number = 8, there should be 8 "ID" lines following the "Species_Num" line. Each ID line has the following format:

```
ID[tab]numerical_id[tab]string_tag
```

String_tag is a tag that is used by the alignments to label species, e.g., hg, mm, canFam, galGal, etc. Numerical_id is a unique number attached to each string_tag. The numerical ID is defined by users. If there are n species, these ids can be selected from 0, 1, 2, …, n-1.

(d) Reference – Numerical ID of the reference species.
(e) Ref_Species – Name of the reference species, e.g., human, mouse, dog, chicken, etc.
(f) Ref_Chr_Size – File that contains the chromosome lengths for reference species.
(g) Comparisons_Num – Number of pairwise comparisons one wish to do to compute conservation scores. If Comparisons_Num=m, the line should be followed by m "COMP" lines.
(h) COMP – Comparisons one wish to do. Each COMP line has the following format:

```
COMP[tab]ID for species 1[tab]ID for species 2
```

The line specifies a pairwise comparison between specie 1 and species 2. Background models will be fit for all pairwise comparisons specified by the "COMP" lines.
(i) BG_Window_Size – Size of the sliding window that is used to compute background identity level.
(j) BG_Sliding_Step – Step size of the sliding window.
(m) Output_Path – Directory to which computed background identity levels will be exported.
(n) Alignment_Suf – Suffix of the alignment files. If the alignment files used to generate conservation scores are ended with *.maf, then Alignment_Suf=.maf
(o) Alignment_File – This species the alignment file that needs to be processed. If there are multiple alignment files, each file should be specified in a separate line

starting with “Alignment_File”. Each line should have the following format:

```
Alignment_File[tab]chr#
```

The program will automatically go to the directory specified by “Alignment_Dir”, find the file named “chr#[Alignment_Suf]”, and process it. For example, if

```
Alignment_File chr1
```

then chr1.maf will be processed.

(3) Run *genome_conservebg* in the following way.
genome_conservebg [parameter file]

For example:
>genome_conservebg genome_conservebg_arg.txt

(4) Prepare a parameter file, say genome_conservecs_arg.txt for conservation score computation. The parameter file is tab-delimited and has the following format:

```
Alignment_Dir /data/genomes/human/hg17/alignment/

Species_Num   8
ID  0   hg
ID  1   panTro
ID  2   mm
ID  3   rn
ID  4   canFam
ID  5   galGal
ID  6   fr
ID  7   danRer

Reference  0
Ref_Species   human
Ref_Chr_Size  /data/genomes/human/hg17/chrlen.txt

Target 0
Tar_Species   human
Tar_Chr_Size  /data/genomes/human/hg17/chrlen.txt

Comparisons_Num 4
COMP   0   2
COMP   0   4
COMP   0   5
COMP   0   7

BG_Window_Size    1000000
BG_Sliding_Step   10000
BG_Path    /data/genomes/human/hg17/conservation/genomelab/bg/
Output_Path   /data/genomes/human/hg17/conservation/genomelab/cs/

Conserve_Window   51
Min_P_Cutoff  1e-4

Alignment_Suf .maf
Alignment_File    chr1
Alignment_File    chr2
Alignment_File    chr3
Alignment_File    chr4
Alignment_File    chr5
Alignment_File    chr6
Alignment_File    chr7
Alignment_File    chr8
Alignment_File    chr9
Alignment_File    chr10
Alignment_File    chr11
Alignment_File    chr12
Alignment_File    chr13
Alignment_File    chr14
Alignment_File    chr15
Alignment_File    chr16
Alignment_File    chr17
Alignment_File    chr18
Alignment_File    chr19
Alignment_File    chr20
Alignment_File    chr21
Alignment_File    chr22
Alignment_File    chrX
Alignment_File    chrY
```

Below are meanings for the parameters.

(a) Alignment_Dir – Path where MAF alignments are stored.

(b) Species_Num – Number of species in the alignments. For example, if the alignments are MULTIZ alignments for 8 vertebrate species, then the species number is 8.

(c) ID – Numerical ID and string tag for each species. If species number = 8, there should be 8 "ID" lines following the "Species_Num" line. Each ID line has the following format:

```
ID[tab]numerical_id[tab]string_tag
```

String_tag is a tag that is used by the alignments to label species, e.g., hg, mm, canFam, galGal, etc. Numerical_id is a unique number attached to each string_tag. The numerical ID is defined by users. If there are n species, these ids can be selected from 0, 1, 2, …, n-1.

(d) Reference – Numerical ID of the reference species.
(e) Ref_Species – Name of the reference species, e.g., human, mouse, dog, chicken, etc.
(f) Ref_Chr_Size – File that contains the chromosome lengths for reference species.
(g) Target – Numerical ID of the target species.
(h) Tar_Species – Name of the target species, e.g., human, mouse, dog, chicken, etc.
(i) Tar_Chr_Size – File that contains the chromosome lengths for target species.
(j) Comparisons_Num – Number of pairwise comparisons one wish to do to compute conservation scores. If Comparisons_Num=m, the line should be followed by m "COMP" lines.
(k) COMP – Comparisons one wish to do. Each COMP line has the following format:

```
COMP[tab]ID for species 1[tab]ID for species 2
```

The line specifies a pairwise comparison between specie 1 and species 2. Conservation scores will be computed from all pairwise comparisons specified by the "COMP" lines.
(l) BG_Window_Size – Size of the sliding window that is used to compute background identity level.
(m) BG_Sliding_Step – Step size of the sliding window.
(n) BG_Path – Directory where background models computed by *genome_conservebg* are stored.
(o) Output_Path – Directory to which computed background identity levels will be exported.
(p) Conserve_Window – Window size for computing conservation scores.
(q) Min_P_Cutoff – Minimal score cutoff.
(r) Alignment_Suf – Suffix of the alignment files. If the alignment files used to generate conservation scores are ended with *.maf, then Alignment_Suf=.maf
(s) Alignment_File – This species the alignment file that needs to be processed. If there are multiple alignment files, each file should be specified in a separate line starting with "Alignment_File". Each line should have the following format:

```
Alignment_File[tab]chr#
```

The program will automatically go to the directory specified by "Alignment_Dir", find the file named "chr#[Alignment_Suf]", and process it. For example, if

```
Alignment_File chr1
```

then chr1.maf will be processed.

(5) Run *genome_conservecs* in the following way.
genome_conservecs [parameter file]

For example:
>genome_conservecs genome_conservecs_arg.txt

(6) After running all the steps above, one should be able to find a collection of *.cs files in the output directory that is specified as “Output_Path” in the parameter file genome_conservecs_arg.txt. The size of each *.cs file (i.e., chr1.cs, chr2.cs, etc.) should be equal to the corresponding chromosome length of the target species.

2.8 Establishing Database Step VI – Creating CDS Indicator

The goal of this step is to create protein coding region indicators. These indicators can be used in downstream analysis such as filtering out coding regions before transcription factor binding motif discovery. The resulting indicators will be stored in *.cds format. A *.cds file is a binary file. Each byte in the file corresponds to a single position in the genome. If the position is located in a protein coding region, the byte is equal to 1, otherwise the byte is equal to 0.

One can generate CDS indicators as follows.

(1) Make sure that *chrlist.txt* and *chrlen.txt* are available in the directory where genome sequences (*.sq files) are stored.

Make sure that a gene annotation file in CisGenome format (i.e., refGene_sorted.txt or refFlat_sorted.txt; refer to section 2.3 and 2.6) is available. This file will be used to specify coding regions.

(2) Run *genome_codingCDS* as follows:

genome_codingCDS -d [Path where genome sequences *.sq are stored] -g [gene structure annotation file] -gt [File format of gene annotations; 0=CisGenome refGene format, 1=CisGenome refFlat format] -s [species] -o [Path where coding region indicators will be stored]

For example:

```
>genome_codingCDS -d /data/genomes/human/hg17/ -g /data/genomes/human/hg17/annotation/refFlat_sorted.txt -gt 1 -s human -o /data/genomes/human/hg17/cds/
```

or

```
>genome_codingCDS -d /data/genomes/human/hg17/ -g /data/genomes/human/hg17/annotation/refGene_sorted.txt -gt 0 -s human -o /data/genomes/human/hg17/cds/
```

(3) After running *genome_codingCDS*, one should be able to find a collection of *.cds files (i.e., chr1.cds, chr2.cds, etc.) in the output directory. The size of each file should be equal to the length of its corresponding chromosome.

3. GENOMICS TOOLBOX II – SEQUENCE AND CONSERVATION SCORE RETRIEVAL

3.1 Introduction

After the local genome database has been established, one can easily retrieve sequences and conservation scores for downstream sequence analysis. This section introduces various functions relevant to this topic.

3.2 List of Functions

(1) *genome_getseq*, *genome_getseq_c* – Retrieving DNA sequences from specified genomic regions. Sequences are retrieved from local genome databases (i.e., *.sq files created in section 2), and the retrieved sequences will be stored in a FASTA file.

(2) *genome_getseqcs*, *genome_getseqcs_c* – Retrieving DNA sequences and conservation scores from specified genomic regions. Sequences and conservation scores are retrieved from local genome databases (i.e., *.sq and *.cs files created in section 2). The retrieved DNA sequences will be stored in a single FASTA file. For each sequence, its corresponding conservation score will be stored in a separate file.

(3) *genome_getmaskedseq*, *genome_getmaskedseq_c* – Retrieving DNA sequences for specified genomic regions. Users can choose to mask protein coding regions or regions with low conservation level by ‘N’.

(4) *genome_getmaskedreg*, *genome_getmaskedreg_c* – Filter genomic regions by their repeat, protein coding and cross-species conservation properties.

(5) *genome_fastaseqmask* – Masking specific motifs from sequences. Both input and output sequences are stored in FASTA files. Masked nucleotides will be replaced by ‘N’ in the output file.

3.3 Sequence Manipulation I – Retrieving Sequences for Specified Genomic Regions.

```
[Usage]
genome_getseq_c -d [Path where genome sequences (*.sq files) are stored]
-s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies
genomic coordinates of target regions] -o [File to save retrieved sequences]
-r [Method to handle +/- strand: assemblybase or genebase]

genome_getseq -d [Path where genome sequences (*.sq files) are stored]
-s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies
genomic coordinates of target regions] -o [File to save retrieved sequences]
-r [Method to handle +/- strand: assemblybase or genebase]
```

genome_getseq_c or *genome_getseq* can be used to retrieve sequences from specified genomic regions.

3.3.1 *genome_getseq_c*

In order to use *genome_getseq_c*, please follow the steps below.

(1) Make sure that relevant local genome databases are available.

(2) Prepare a tab-delimited text file that contains genomic regions for which one wants to get sequences. The file should have the following format.

```
Sequence1_ID[tab]chromosome[tab]start[tab]end[tab]strand
Sequence2_ID[tab]chromosome[tab]start[tab]end[tab]strand
…
```

In this file, “Sequence_ID” is a unique numerical or string ID for each region. “Chromosome” is the chromosome name (e.g., chr1, chr2, chrX, chrY, etc.). “Start” and “End” are integers that specify genomic coordinates of a region. Both “Start” and “End” are zero-based, i.e., the first position in each chromosome is indexed by 0, the second position indexed by 1, and so on. “Strand” specifies if a region is located in the forward or the backward strand of the assembly. Below is a sample file Gli_coordinates.txt:

```
0   chr13  60949397   60950371   +
1   chr2   146644626  146645570  +
2   chr13  60951377   60952953   -
3   chr12  53347310   53348354   +
4   chrX   52789693   52789967   -
5   chr12  53340519   53341015   +
6   chr11  77915220   77915560   +
```

(3) Run *genome_getseq_c*. For example:

```
>genome_getseq_c –d /data/genomes/mouse/mm6/ -s mouse –i Gli_coordinates.txt –o Gli.fa –r assemblybase
```

In this example, DNA sequences for target regions will be extracted from mouse mm6 assembly, and will be saved to a file named “Gli.fa”.

If “–r” option is “assemblybase”, then all sequences will be extracted from the “+” strand of the assembly.

If “–r” option is “genebase”, sequences will be first extracted from the “+” strand of the assembly, the extracted sequences will then be adjusted according to their strand information (the 5th column in Gli_coordinates.txt). If a region is “-“, then the reverse complement of the extracted sequence will be saved to the output file; if a region is “+”, then the extracted sequence will be saved directly to the output file.

(4) After Running *genome_getseq_c*, one should get a file (Gli.fa in the above example) where extracted sequences are saved in FASTA format.

3.3.2 *genome_getseq*

genome_getseq can be used in a similar way as *genome_getseq_c*. The only difference between these two functions is the input coordinates file. For *genome_getseq*, chromosomes (the 2nd column in the coordinates file) should be specified by their numerical IDs (e.g., 1, 2, …, 23, 24 for human) instead of a string (e.g., chr1, chr2, …, chrX, chrY for human). Below is a sample coordinate file (Gli_coordinates_n.txt) for genome_getseq. The file specifies the same mouse regions as Gli_coordinates.txt in section 3.3.1.

```
0   13  60949397   60950371   +
1   2   146644626  146645570  +
2   13  60951377   60952953   -
3   12  53347310   53348354   +
4   20  52789693   52789967   -
5   12  53340519   53341015   +
6   11  77915220   77915560   +
```

One can run *genome_getseq* as below:

```
>genome_getseq –d /data/genomes/mouse/mm6/ -s mouse –i Gli_coordinates_n.txt –o Gli.fa –r assemblybase
```

3.4 Sequence Manipulation II – Retrieving Sequences and Conservation Scores for Specified Genomic Regions

[Usage]

genome_getseqcs_c -d [Path where genome sequences (*.sq files) are stored] -c [Path where genome conservation scores (*.cs files) are stored] -s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies genomic coordinates of target regions] -o [Output Directory] -a [Output File Title] -r [Method to handle +/- strand: *assemblybase* or *genebase*] -f [Type of output files to save conservation scores: *cs*, *bed*, *txt*]

genome_getseqcs -d [Path where genome sequences (*.sq files) are stored] -c [Path where genome conservation scores (*.cs files) are stored] -s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies genomic coordinates of target regions] -o [Output Directory] -a [Output File Title] -r [Method to handle +/- strand: *assemblybase* or *genebase*] -f [Type of output files to save conservation scores: *cs*, *bed*, *txt*]

genome_getseqcs_c and *genome_getseqcs* are extensions of *genome_getseq_c* and *genome_getseq*. These two functions can be used to retrieve sequences and conservation scores simultaneously from specified genomic regions.

3.4.1 *genome_getseqcs_c*

In order to use *genome_getseqcs_c*, please follow the steps below.

(1) Make sure that the local genome database is available. The database should contain both sequences (*.sq files) and conservation scores (*.cs files).

(2) Prepare a tab-delimited text file that contains genomic regions for which one wants to get sequences and conservation scores. The file should have the following format.

```
Sequence1_ID[tab]chromosome[tab]start[tab]end[tab]strand
Sequence2_ID[tab]chromosome[tab]start[tab]end[tab]strand
…
```

For example, Gli_coordinates.txt:

```
0   chr13  60949397   60950371   +
1   chr2   146644626  146645570  +
2   chr13  60951377   60952953   -
3   chr12  53347310   53348354   +
4   chrX   52789693   52789967   -
5   chr12  53340519   53341015   +
6   chr11  77915220   77915560   +
```

In this file, the first column specifies a unique ID for each region. The ID could be a number or a string. The second column is chromosome name (e.g., chr1, chr2, chrX, chrY, etc.). The third and fourth columns specify region starts and ends. Both are integers and are zero-based, i.e., the first position in each chromosome is indexed by 0, the second position indexed by 1, and so on. The fifth column specifies the strand information of a region (i.e., + or – in relative to the assembly).

(3) Run *genome_getseqcs_c* as below:

```
>genome_getseqcs_c –d /data/genomes/mouse/mm6/ –c /data/genomes/mouse/mm6/conservation/phastCons/ -s mouse –i Gli_coordinates.txt –o /users/output/ -a Gli –r assemblybase –f cs
```

In this example, DNA sequences and conservation scores for regions specified in Gli_coordinates.txt will be extracted from mouse genome (assembly mm6). The retrieved sequences will be saved to the file /users/output/Gli.fa. For each sequence, a separate file /users/output/Gli_[index]_[ID].[suffix] will be generated to save the corresponding conservation scores. Here, [index] is a numerical ID automatically attached to each sequence. The first sequence is indexed as 0, the second sequence is indexed as 1, and so on. [ID] is the numerical or string ID specified in the first column of Gli_coordinates.txt. [suffix] is the file type specified by "-f" option (in this example, *cs*). As a result, one can find conservation scores for region chr13:60949397-60950371 in a file named /users/output/Gli_0_0.cs, and find conservation scores for region chr2:146644626-146645570 in a file named /users/output/Gli_1_1.cs.

"-r" specifies strand type. If "–r assemblybase", then all sequences and conservation scores will be extracted from the "+" strand of the assembly. If "–r genebase", sequences and conservation scores will be first extracted from the "+" strand of the assembly, the extracted sequences and conservation scores will then be adjusted according to their strand information (the 5th column in Gli_coordinates.txt). If a region is "-", then the reverse complement of the extracted sequence will be saved to the output FASTA file, and the reverted conservation scores will be saved to the output score file; if a region is "+", then the extracted sequences and conservation scores will be saved directly to the output file.

"-f" specifies the file format to save conservation scores. "-f cs" means that conservation scores will be saved in *.cs files. As have been discussed in section 2.3, *.cs files are binary files. Each byte in the file corresponds to a single position in the DNA sequence. The score ranges from 0 to 255. The bigger the more conserved. "-f txt" means that conservation scores will be saved in *.txt files. *.txt files are ASCII files. Each line in the file corresponds to a single position in the DNA sequence. The score ranges from 0 to 255. "-f bed" means that conservation scores will be saved in *.bed files. *.bed files has the BED format (http://genome.ucsc.edu/goldenPath/help/customTrack.html). Each line in the file corresponds to a single position in the DNA sequence. The score is linearly converted from interval [0, 255] to interval [0, 1000]. A *.bed file can be visualized in UCSC Genome Browser as a customer track. One can use this file to check visually if conservation scores are correctly extracted.

Hint: In the process of establishing local genome database, users can use the function here to check if Footprint scores or GenomeLab conservation scores are correctly computed.

3.4.2 *genome_getseqcs*

genome_getseqcs can be used in a similar way as *genome_getseqcs_c*. The only difference between these two functions is the input coordinates file. For *genome_getseqcs*, chromosomes (the 2nd column in the coordinates file) should be specified by their numerical IDs (e.g., 1, 2, …, 23, 24 for human) instead of a string (e.g., chr1, chr2, …, chrX, chrY for human). Below is a sample coordinate file (Gli_coordinates_n.txt) for genome_getseqcs. The file specifies the same mouse regions as Gli_coordinates.txt in section 3.4.1.

```
0   13  60949397   60950371   +
1   2   146644626  146645570  +
2   13  60951377   60952953   -
3   12  53347310   53348354   +
4   20  52789693   52789967   -
5   12  53340519   53341015   +
6   11  77915220   77915560   +
```

One can run *genome_getseqcs* as below:

```
>genome_getseqcs_c –d /data/genomes/mouse/mm6/ –c /data/genomes/mouse/mm6/conservation/phastCons/ -s mouse –i Gli_coordinates_n.txt –o /users/output/ -a Gli –r assemblybase –f cs
```

3.5 Sequence Manipulation III – Retrieving Sequences in which Protein Coding Regions and/or Non-Conserved Regions are Masked.

```
[Usage]
genome_getmaskedseq_c -d [Path where genome sequences (*.sq files) are stored] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored] -cds [Path where protein coding indicators (*.cds files) are stored] -s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies genomic coordinates of target regions] -o [Output File] -r [Method to handle +/- strand: assemblybase or genebase]

genome_getmaskedseq -d [Path where genome sequences (*.sq files) are stored] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored] -cds [Path where protein coding indicators (*.cds files) are stored] -s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies genomic coordinates of target regions] -o [Output File] -r [Method to handle +/- strand: assemblybase or genebase]
```

genome_getmaskedseq_c or *genome_getmaskedseq* can be used to retrieve sequences for specific genomic regions. Protein coding regions and/or regions with low conservation scores can be masked by 'N' in the retrieved sequences.

3.5.1 *genome_getmaskedseq_c*

In order to use *genome_getmaskedseq_c*, please follow the steps below.

(1) Make sure that relevant local genome databases are available.

(2) Prepare a file in COD_C format (Appendix A.1.6) that specifies target genomic regions. Sequences will be extracted from the specified regions.

(3) Run *genome_getmaskedseq_c*. For example:

```
>genome_getmaskedseq_c -d /data/genomes/human/hg17/ -c 40 -cd /data/genomes/human/hg17/conservation/phastcons/ -cds /data/genomes/human/hg17/cds/ -s human -i coordinates.txt -o seq.fa -r assemblybase
```

In *genome_getmaskedseq_c*,

"-d" specifies the directory where genome sequences (*.sq files) are stored. Make sure that the directory contains two other files: chrlist.txt and chrlen.txt.

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

"-c" specifies the conservation cutoff. Once "-c" and "-cd" are specified, sequences whose conservation scores are less than the cutoff will be masked by "N". If one does not want to apply the conservation filter, please do not include "–c" and "-cd" options in the command.

"-cds" specifies the directory where protein coding indictors (*.cds files) are stored. Once “-cds” is specified, protein coding regions will be masked by “N”. If one does not want to apply the CDS filter, please do not include “-cds” option in the command.

“-s” specifies species name, e.g., human, mouse, dog, cow, chicken, zebrafish.

“-i” specifies the coordinates file. The file must be in COD_C format.

“-o” specifies the output file. Retrieved sequences will be saved to this file in FASTA format.

“-r” specifies whether reverse complement sequences should be returned according to the strand information (column 5) in coordinates file.

If “-r assemblybase”, then all sequences will be extracted from the “+” strand of the genome assembly. The masked sequences will be saved directly to the output FASTA file.

If “–r genebase”, then sequences will be first extracted from the “+” strand of the assembly. Next, the masked sequences will be adjusted according to their strand information (column 5 in coordinates file). If a region is “-“, then the reverse complement of the masked sequence will be saved to the output FASTA file; if a region is “+”, then the masked sequences will be saved directly to the output file.

(4) After running *genome_getmaskedseq_c*, one should be able to find an output file specified by the “-o” option. The file contains masked sequences in FASTA format. In the output file, all repeats (small letters “a”, “c”, “g” and “t”) in the genome assembly are masked by “N”. Protein coding regions and regions with low conservation scores are masked by “N” when specified by users.

3.5.2 *genome_getmaskedseq*

genome_getmaskedseq can be used in a similar way as *genome_getmaskedseq_c*. The only difference is that the coordinates file used for *genome_getmaskedseq* must be in COD_N (Appendix A.1.7) format instead of COD_C format.

Below is an example to run *genome_getmaskedseq*:

```
>genome_getmaskedseq -d /data/genomes/human/hg17/ -c 40 -cd
/data/genomes/human/hg17/conservation/phastcons/ -cds
/data/genomes/human/hg17/cds/ -s human -i coordinates_n.txt -o seq.fa -r
assemblybase
```

3.6 Sequence Manipulation IV – Filtering Genomic Regions by Repeats, Conservation Scores and CDS Status.

```
[Usage]
genome_getmaskedreg_c -d [Path where genome sequences (*.sq files) are stored] -r [Non-repeat ratio] -c [Conservation cutoff] -cr [Conservation ratio] -cd [Path where genome conservation scores (*.cs files) are stored] -cds [Non-CDS ratio] -cdsd [Path where protein coding indicators (*.cds files) are stored] -s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies genomic coordinates of target regions] –o [Output File]

genome_getmaskedreg -d [Path where genome sequences (*.sq files) are stored] -r [Non-repeat ratio] -c [Conservation cutoff] -cr [Conservation ratio] -cd [Path where genome conservation scores (*.cs files) are stored] -cds [Non-CDS ratio] -cdsd [Path where protein coding indicators (*.cds files) are stored] -s [Species name, e.g., mouse, human, dog, etc.] -i [File that specifies genomic coordinates of target regions] –o [Output File]
```

genome_getmaskedreg_c or *genome_getmaskedreg* can be used to filter genomic regions according to their repeat, protein coding and phylogenetic conservation characteristics. Sequences whose conservation score >= [Conservation cutoff] are defined as conserved base pairs. A genomic region will be filtered out if one of the following conditions is satisfied:

(i) The fraction of non-repeat base pairs < [Non-repeat ratio];

(ii) The fraction of conserved base pairs < [Conservation ratio];

(iii) The fraction of non-protein-coding region < [Non-CDS ratio].

One can choose to apply part or all of the filters. Regions that can pass the filtering will be saved to the output file.

3.6.1 *genome_getmaskedreg_c*

In order to use *genome_getmaskedreg_c*, please follow the steps below.

(1) Make sure that relevant local genome databases are available.

(2) Prepare a file in COD_C format (Appendix A.1.6) that specifies input genomic regions.

(3) Run *genome_getmaskedreg_c*. For example:

```
> genome_getmaskedreg_c -d /data/genomes/human/hg17/ -r 0.9 -c 40 -cr 0.9 -cd /data/genomes/human/hg17/conservation/phastcons/ -cds 0.9 -cdsd /data/genomes/human/hg17/cds/ -s human -i /data/genomes/human/hg17/coordinates.txt -o coordinates_masked.txt
```

In *genome_getmaskedreg_c*,

"-d" specifies the directory where genome sequences (*.sq files) are stored. Make sure that the directory contains two other files: chrlist.txt and chrlen.txt.

"-r" specifies the non-repeat ratio. If one does not want to apply the repeat filter, please do not include the "–r" option in the command.

"-c" specifies the conservation cutoff. Base pairs whose conservation scores >= conservation cutoff are defined as conserved base pairs.

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

"-cr" specifies the conservation ratio. If one does not want to apply the conservation filter, please do not include the "–c", "-cd" and "-cr" options in the command.

"-cdsd" specifies the directory where protein coding indictors (*.cds files) are stored.

"-cds" specifies the non-CDS ratio. If one does not want to apply the CDS filter, please do not include the "–cds" and "-cdsd" options in the command.

"-s" specifies species name, e.g., human, mouse, dog, cow, chicken, zebrafish.

"-i" specifies the input coordinates file. The file must be in COD_C format.

"-o" specifies the output file. The file will save coordinates of regions that can pass the filtering. Coordinates will be saved in COD_C format.

3.6.2 *genome_getmaskedreg*

genome_getmaskedreg can be used in a similar way as *genome_getmaskedreg_c*. The only difference is that the coordinates file used for *genome_getmaskedreg* must be in COD_N (Appendix A.1.7) format instead of COD_C format, and the output file of *genome_getmaskedreg* will be in COD_N format too.

Below is an example to run *genome_getmaskedreg*:

```
> genome_getmaskedreg -d /data/genomes/human/hg17/ -r 0.9 -c 40 -cr 0.9
-cd  /data/genomes/human/hg17/conservation/phastcons/  -cds  0.9  -cdsd
/data/genomes/human/hg17/cds/          -s          human          -i
/data/genomes/human/hg17/coordinates_n.txt -o coordinates_masked.txt
```

3.7 Sequence Manipulation V – Masking Specified Regions From FASTA Sequences.

```
[Usage]
genome_fastaseqmask -i [Input FASTA file] -m [Mask file] -mt [Type of masking: 0-apply soft masking; 1- apply hard masking] -o [Output FASTA file]
```

genome_fastaseqmask can be used to mask specified regions from sequences stored in FASTA format. This function is useful when one wants to mask certain low-complexity motifs before conducting de novo motif discovery.

In order to use *genome_fastaseqmask*, please follow the steps below.

(1) Prepare a file that contains sequences to be processed in FASTA format.

(2) Prepare a file that specifies what regions in the input sequences need to be masked. The file must be in COD_FA format, e.g.:

```
0      3187   3199   -      CGCGCGCGCTGCC   0.608000
0      3903   3915   +      CGGGCCTCCTCCC   0.282000
1      3999   4011   +      TCCTCGCCCTGCG   0.834000
1      7666   7678   +      CCCGCAGGCTGCG   0.508000
2      8252   8264   +      TGCGTCCGCTCCG   0.666000
3      9026   9038   +      CCGTGGCTCGGCC   0.258000
3      9141   9153   -      CCGGCGCCCGGCC   0.342000
3      9603   9615   +      CCGGTCCCCTCCC   0.288000
```

(3) Run *genome_fastaseqmask*. For example:

```
> genome_fastaseqmask -i input.fa -m mask.txt -mt 1 -o output.fa
```

In *genome_fastaseqmask*,

"-i" specifies the FASTA file that contains input sequences.

"-m" specifies the COD_FA file that contains all regions that need to be masked.

"-mt" specifies the mask type. If "-mt 0", masked sequences will be converted from capital letters "A", "C", "G", "T" to small letters "a", "c", "g" and "t". If "mt 1", masked sequences will be converted to "N".

"-o" specifies the output file that will save masked sequences in FASTA format.

4. GENOMICS TOOLBOX III –ANNOTATING SPECIFIED REGIONS

4.1 Introduction

Besides sequence retrieval, genomics toolbox can also be used to retrieve other related information for specified genomic regions. This section introduces various functions relevant to this topic.

4.2 List of Functions

(1) *genome_getcsgcsummary* – Get nucleotide occurrence frequencies and conservation score distribution for specified genomic regions.

(2) *refgene_getnearestgene* – Associating genomic regions with neighboring genes.

(3) *refgene_gettssaround* – Get regions surrounding transcription start sites.

(4) *refgene_getaffy* – Get Affymetrix probeset IDs for a set of genes.

(5) *reflex_getmultiortholog, refgene_getmultiortholog* – Search for orthologs of a set of genes.

4.3 Get Nucleotide Occurrence Frequencies and Conservation Score Distribution for Specified Genomic Regions

```
[Usage]
genome_getcsgcsummary -gd [Path where genome sequences (*.sq files) are stored] -i [File that specifies genomic coordinates of target regions] -o [Output file title] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored]
```

genome_getcsgcsummary can be used to count occurrence frequencies of A, C, G and T, and to get the distribution of conservation scores for specified genomic regions. After specifying the coordinates of genomic regions in COD_C format, the program will automatically check the non-repeat regions. Occurrence frequencies of A, C, G, and T will be returned in a text file named [Output file title]_gc.stat. The number of non-repeat base pairs at each conservation score level (i.e., 0, 1, 2, …, 255) will be counted, and the resulting histogram will be saved to a text file named [Output file title]_cs.stat. Optionally, users can also specify a conservation cutoff. If the conservation cutoff is given, nucleotide occurrence frequencies and conservation score distributions will only be counted for sequences where the conservation scores >= [Conservation cutoff].

genome_getcsgcsummary can be used as follows.

(1) Make sure that relevant local genome databases are available.

(2) Prepare a file in COD_C format that specifies target genomic regions to be studied (Appendix A.1.6), e.g., input_coord.txt.

(3) Run *genome_getcsgcsummary*. For example:

```
> genome_getcsgcsummary -gd /data/genomes/human/hg17/ -i input_coord.txt -o output_summary -c 40 -cd /data/genomes/human/hg17/conservation/phastcons/
```

In *genome_getcsgcsummary*,

"-gd" specifies the directory where genome sequences (*.sq files) are stored. Make sure that the directory contains two other files: chrlist.txt and chrlen.txt.

"-i" specifies the file that contains target genomic regions in COD_C format.

"-o" specifies the title of the output file.

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

"-c" specifies the conservation cutoff. If one does not want to set a conservation cutoff, and if one wants to get summary statistics for all non-repeat base pairs, please do not include "-c" and "-cd" options in the command.

(4) Check [Output file title]_gc.stat and [Output file title]_cs.stat for results. In the above example, [Output file title] is "output_summary". The *_cs.stat file contains 256 lines. Line 1 counts how many non-repeat base pairs have a conservation score 0;

line 2 counts how many non-repeat base pairs have a conservation score 1, etc.

4.4 Associating Genomic Regions with Neighboring Genes

```
[Usage]
refgene_getnearestgene -d [Path of the gene annotation file] -dt [Format
of the gene annotation file] -s [Species name, e.g., human, mouse, dog]
-i [File that specifies genomic regions to be annotated] -o [Output file]
-r [Definition of Distance] -up [Maximum upstream distance allowed] -down
[Maximum downstream distance allowed]
```

refgene_getnearestgene can be used to annotate genomic regions using their neighboring genes. Users need to define a distance measure first. Based on the distance measure, the gene that is closest to each genomic region will be reported. Users also need to specify two distance limits, i.e., [Maximum upstream distance allowed] and [Maximum downstream distance allowed], if the distance to the closed gene is greater than the limits, the genomic region will not be annotated.

Assume that the mid point of a genomic region is *x*, TSS = transcription start, TES = transcription end, CDSS = protein coding region start, CDSE = protein coding region end, [Maximum upstream distance allowed]=M_u, [Maximum downstream distance allowed]= M_d. Users can choose from one of the distance measures below.

(1) 0: TSS-up, TES-down. This is the distance to a gene. If *x* falls within the interval specified by TSS and TES, then the distance = 0. If *x* is located upstream of TSS, then the distance = |*x*-TSS|. If x is located downstream of TES, then the distance = |*x*-TES|. If *x* is located upstream of TSS and is >M_u base pairs away from TSS, or if *x* is located downstream of TES and is >M_d base pairs away from TES, then the region will not be annotated.

(2) 1: TSS-up, TSS-down. This is the distance to TSS. The distance = |*x*-TSS |. If *x* is located upstream of TSS and is >M_u base pairs away from TSS, or if *x* is located downstream of TSS and is >M_d base pairs away from TSS, then the region will not be annotated.

(3) 2: TES-up, TES-down. This is the distance to TES. The distance = |*x*-TES |. If *x* is located upstream of TES and is >M_u base pairs away from TES, or if *x* is located downstream of TES and is >M_d base pairs away from TES, then the region will not be annotated.

(4) 3: CDSS-up, CDSE-down. This is the distance to protein coding regions. If *x* falls within the interval specified by CDSS and CDSE, then the distance = 0. If *x* is located upstream of CDSS, then the distance = |*x*- CDSS |. If x is located downstream of CDSE, then the distance = |*x*- CDSE|. If *x* is located upstream of CDSS and is >M_u base pairs away from CDSS, or if *x* is located downstream of CDSE and is >M_d base pairs away from CDSE, then the region will not be annotated.

(5) 4: CDSS-up, CDSS-down. This is the distance to CDSS. The distance = |*x*-CDSS |. If *x* is located upstream of CDSS and is >M_u base pairs away from CDSS, or if *x* is located downstream of CDSS and is >M_d base pairs away from CDSS, then the region will not be annotated.

(6) 5: CDSE -up, CDSE -down. This is the distance to CDSE. The distance = |*x*-

CDSE |. If *x* is located upstream of CDSE and is >M_u base pairs away from CDSE, or if *x* is located downstream of CDSE and is >M_d base pairs away from CDSE, then the region will not be annotated.

refgene_getnearestgene can be used as follows.

(1) Make sure that local genome databases contain relevant gene structure annotations in either REFGENE_CISGENOME (Appendix A.1.17) or REFFLAT_CISGENOME (Appendix A.1.19) format.

(2) Prepare a file in COD_C format (Appendix A.1.6) that specifies target genomic regions that need to be annotated, e.g., target_coord.txt.

(3) Run *refgene_getnearestgene*. For example:

```
> refgene_getnearestgene -d
/data/genomes/human/hg17/annotation/refFlat_sorted.txt –dt 1 –s human -i
target_coord.txt -o target_annot.txt –r 0 –up 50000 –down 50000
```

In *refgene_getnearestgene,*

"-d" specifies the whole genome gene annotation file. This file should be either in REFGENE_CISGENOME format or in REFFLAT_CISGENOME format.

"-dt" specifies the format of whole genome gene annotation file. If "-dt 0", then the annotation file specified by "-d" should be in REFGENE_CISGENOME format. If "-dt 1", then the annotation file should be in REFFLAT_CISGENOME format.

"-s" specifies the species name, e.g., human, mouse, dog, cow, chicken, zebrafish.

"-i" specifies the file that contains target genomic regions in COD_C format.

"-o" specifies the the output file.

"-r" specifies what distance measure to use. One can choose from 0 to 5.

"-up" specifies the [Maximum upstream distance allowed].

"-down" specifies the [Maximum downstream distance allowed].

(4) After running *refgene_getnearestgene,* the results will be saved to a tab-delimited output file specified by "-o" option. In the output file, each region is linked to its closest gene (if there is any), and the gene information is provided in REFGENE_CISGENOME or REFFLAT_CISGENOME format.

4.5 Getting Regions Surrounding Transcriptional Starts

```
[Usage]
refgene_gettssaround -d [Path of the gene annotation file] -t [File that specifies input RefSeq IDs] -o [Output File] -s [Species name, e.g., human, mouse, dog] -up [Upstream window size] -down [Downstream window size] -c [Chromosome Lengths]
```

refgene_gettssaround can be used to get genomic regions surrounding transcriptional starts (TSS) for specified genes. After specifying a list of RefSeq IDs and how many base pairs upstream and downstream from TSS one wants to extend, the program will return a file in COD_C format that contains regions surrounding TSS for all the specified genes.

refgene_gettssaround can be used as follows.

(1) Make sure that local genome databases contain relevant gene structure annotations in REFGENE_CISGENOME (Appendix A.1.17) format.

(2) Prepare a text file that contains RefSeq IDs for all genes one wants to study. Each line contains a single RefSeq ID. For example, genelist.txt:

```
NM_001633
NM_005952
NM_000638
NM_000295
NM_005952
```

(3) Run *refgene_gettssaround*. For example:

```
> refgene_gettssaround -d /data/genomes/human/hg17/annotation/refGene_sorted.txt -t genelist.txt -o genetss5k1k_coord.txt -s human -up 5000 -down 1000 -c /data/genomes/human/hg17/chrlen.txt
```

In *refgene_gettssaround,*

“-d” specifies the whole genome gene annotation file. This file should be in REFGENE_CISGENOME format.

“-t” specifies the file that contains target RefSeq IDs.

“-s” specifies the species name, e.g., human, mouse, dog, cow, chicken, zebrafish.

“-o” specifies the output file.

“-up” specifies how many base pairs upstream of TSS one want to extract.

“-down” specifies how many base pairs downstream of TSS one want to extract.

“-c” specifies the file that contains chromosome lengths for all chromosomes. Such a file is usually generated automatically when the local genome database is established.

(4) After running *refgene_gettssaround*, the extracted region will be saved to the output file in COD_C format, which can then be used as input for other functions such as *genome_getseq*, *motifmap_matrixscan_genome*, etc.

4.6 Get Affymetrix Probeset IDs for Specified Genes

```
[Usage]
refgene_getaffy -d [File that contains the mapping from probeset ID to
RefSeq ID] -i [File that specifies input RefSeq IDs] -c [Column number
of RefSeq IDs] -o [Output File]
```

refgene_getaffy can be used to retrieve probeset IDs from Affymetrix expression arrays for a set of specified genes. After specifying a list of RefSeq IDs and a mapping from probeset ID to RefSeq ID, the program will return a file that contains probeset IDs associated with the given genes..

refgene_getaffy can be used as follows.

(1) Prepare a text file that contains RefSeq IDs for all genes one wants to study. The file is tab-delimited, and the RefSeq IDs should be in the same column, e.g.,

```
NM_001633
NM_005952
NM_000638
NM_000295
NM_005952
```

Or:

```
Gene1	NM_001633
Gene2	NM_005952
Gene3	NM_000638
Gene4	NM_000295
Gene5	NM_005952
```

(2) Prepare a text file that contains a mapping from probeset IDs to RefSeq IDs. The file has the following format:

```
1415703_at	NM_021523
1415705_at	---
1415706_at	NM_009938
1415707_at	NM_175300
1415708_at	NR_002321 /// NR_002322
```

Each line contains two columns separated by a tab:
Probeset_ID[tab]RefSeq_ID

If a probeset is associated with multiple RefSeq IDs, then RefSeq IDs are separated by “[space]///[space]”. If a probeset has no associated RefSeq ID, then use “---” to represent the RefSeq ID.

The mapping can be obtained from Affymetrix's *.csv annotation files for each individual array types. To get the *.csv annotation, go to http://www.affymetrix.com/support/technical/byproduct.affx?cat=exparrays, choose the array type you are using, and download the corresponding "NetAffx Annotation Files" with *.csv suffix. Open it using Excel, cut the column "Probe Set" and "RefSeq Transcript ID", paste them to a new excel file, remove the first line (i.e. "Probe Set" and "RefSeq Transcript ID"), and save it as a tab-delimited text file. The resulting file can then be used as the mapping file.

(3) Run *refgene_getaffy*. For example:

```
>refgene_getaffy -d Mouse430_2_affy2refid.txt -i genelist.txt -c 0 -o
genelist_affy.txt
```

In *refgene_getaffy*,

"-d" specifies the file that contains the mapping from probeset ID to RefSeq ID.

"-i" specifies the input file that contains target RefSeq IDs.

"-c" specifies in which column of the input file are the RefSeq IDs. The column is indexed from 0, i.e., the first column is 0, the second column is 1, etc.

"o" specifies the output file.

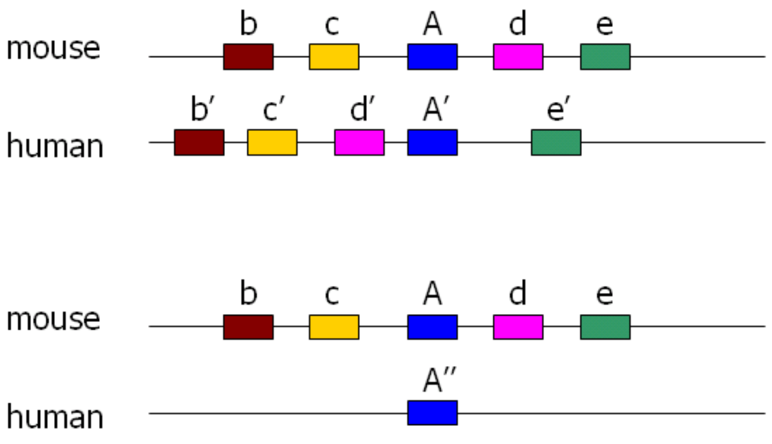
(4) After running *refgene_getaffy*, the results will be saved to a tab-delimited output file which is specified by "-o" option. The first column is the probeset ID, and the remaining columns are copied from the input file.

4.7 Identifying Ortholog (Homolog) Genes

```
[Usage]
refflex_getmultiortholog -i [Input file] -c [RefGene starting column] -d
[File that specifies annotation databases] -o [Output file title]

refgene_getmultiortholog  -i  [Input  file]  -d  [File  that  specifies
annotation databases] -o [Output file title]
```

refflex_getmultiortholog and *refgene_getmultiortholog* can be used to identify orthologs for a list of genes. The orthologs are identified using an *ad hoc* co-linearity criterion. We use the following example to illustrate the basic idea. In order to identify the human ortholog of a mouse gene A, all alignments of the mouse gene to the human genome will be extracted from human annotation database (mouseRefGene_Sorted.txt). For illustrations purpose, let us assume that two such hits can be found, denoted by A' and A'' respectively (see the Figure below).



For the mouse gene A, four flanking genes (two on each side) are extracted. The four genes are picked up one by one starting from the mouse gene closed to A. The four flanking genes are required to be non-overlapping with A as well as with each other. Any flanking genes more than 3Mb away from gene A are then excluded from further analysis.

On the other hand, each hit in human, say A', is also extended to both sides until the extension on each side can cover 10 non-overlapping flanking genes or the extension reaches 3Mb, whichever comes first. The extended region is called as a human candidate region. Mouse gene A and its four flanking genes are then compared with all genes within the human candidate regions. For each human candidate region (i.e., A' or A''), we count how many of the mouse genes (gene A and flanking genes b, c, d, e) can find a match in the candidate region. We denote this number by M. In our example, M=5 for human candidate region A', and M=1 for human candidate region A''.

For each gene pair (excluding gene A) that can be found both in mouse and in human, we check if their relative position has been changed. A gene pair is defined to be a collinear gene pair, if (i) two genes in the gene pair are on the same side of gene A (e.g., b-c) and are still on the same side of gene A's human homolog (i.e., A' or A''), or if (ii) the two genes in a gene pair are on different sides of gene A (e.g., b-e) and are still on different sides of gene A's human homolog. For each human candidate region, we count how many collinear gene pairs one can find. We denote this number by O. In our example, O=3 for human candidate region A', and O=0 for human candidate region A''. The 3 co-linear gene pairs in A' region are: b'-c', b'-e', c'-e'.

Next, we compute a co-linearity score for each human candidate region as Score=M+1.2*O. We then rank all human candidate regions by the score. The region with the highest score will be selected as the ortholog region. In our example, A' is therefore picked up as the human ortholog to mouse gene A.

When implementing this algorithm to handle multiple species, we usually choose a link species first to facilitate making judgment of whether a gene has a match in another species or not. RefSeqs from the link species are mapped to all species. As a result, all genes including A, A', A'', b, c, d, e, b', c', d', e' are indeed RefSeqs from the link species aligned to different genomes. In this way, a match between two species can be easily judged by comparing the RefSeq IDs. If the input RefSeq IDs are from the original species, we will first obtain RefSeq IDs from the link species that overlap them best in the alignment. The link species RefSeq IDs will then be used to identify orthologs. The most commonly used link species is mouse. The alignment of mouse RefSeqs to different genomes can be easily obtained by *refgene_getspeciesspecific* when establishing the local genome databases.

4.7.1 *refflex_getmultiortholog*

refflex_getmultiortholog can be used to search for orthologs if one provides both RefSeq IDs and gene structure annotations for genes to be studied.

(1) Prepare a parameter file, say, orthologsetting.txt that specifies the annotation database to be used. The file should be in GENOME_ANNOTATION_SETTING format (Appendix A.1.23).

(2) Prepare a tab-delimited file, say, inputgene.txt that specifies the genes for which one wants to retrieve orthologs. The file should contain structures of the gene in REFGENE_CISGENOME_FLEXIBLE format (Appendix A.1.20).

(3) Run *refflex_getmultiortholog*. For example,

```
> refflex_getmultiortholog -i inputgene.txt -c 0 -d orthologsetting.txt -o inputgene_ortholog
```

In *refflex_getmultiortholog,*

"-i" specifies the file that contains genes to be studied.

"-c" specifies from which column the gene structure annotation in REFGENE_CISGENOME format will start in the input file. The columns are tab-delimited and indexed from 0. The first column is indexed as 0, the second column is indexed as 1, and so on.

"-d" specifies the file that contains links to annotation databases.

“-o” specifies the output file title.

(4) After running *refflex_getmultiortholog*, check a file named *.nsomap for results. For example, if the output file title is *inputgene_ortholog*, then the orthologs found will be saved to *inputgene_ortholog.nsomap*. The output file is in REFGENE_ORTHOLG format (Appendix A.1.21).

4.7.2 *refgene_getmultiortholog*

refgene_getmultiortholog can be used to search for orthologs if one only provides RefSeq IDs for genes to be studied.

(1) Prepare a parameter file, say, orthologsetting.txt that specifies the annotation database to be used. The file should be in GENOME_ANNOTATION_SETTING format.

(2) Prepare a text file, say, inputgene.txt. Each line in the file specifies a RefSeq ID.

(3) Run *refgene_getmultiortholog*. For example,

```
> refgene_getmultiortholog -i inputgene.txt -d orthologsetting.txt -o inputgene_ortholog
```

In *refgene_getmultiortholog*,
“-i” specifies the file that contains genes to be studied.
“-d” specifies the file that contains links to annotation databases.
“-o” specifies the output file title.

(4) After running *refgene_getmultiortholog*, check a file named *.nsomap for results. For example, if the output file title is *inputgene_ortholog*, then the orthologs found will be saved to *inputgene_ortholog.nsomap*. The output file is in REFGENE_ORTHOLG format.

5. Microarray Toolbox – Microarray Gene Selection

5.1 Introduction

Selecting genes that show specific spatial or temporal expression patterns is a routine analysis to identify key players in a biological system. The microarray gene selection tool, PowerExpress, is designed to handle this task.

5.2 List of Functions

(1) *powexpress* – Selecting genes that show specific expression patterns.

(2) *powexpress_getspecificprobe* – Get raw expression data for specified probesets.

(3) *powexpress_getnrprobe* – Remove redundant probesets from the raw data file.

5.3 Selecting Genes that Show Specific Expression Patterns

```
[Usage]
> powexpress -d [datafile] -a [annotation file] -c [parameter file] -o
[output file]
```

powexpress is a tool for selecting genes from microarrays. The main features of PowerExpress include:

(1) *powexpress* can be used to search for genes that show complex expression patterns of interest (e.g. genes that show specific temporal or spatial patterns). In *powexpress*, one can specify a complex criteria for conducting multiple sample comparisons such as "(mutant 1 < wild type < mutant 2) and [(mutant 3 > wild type) or (mutant 4 <mutant 5)]", and the program will then automatically rank the genes according to the specified criteria.

(2) *powexpress* is especially useful for dealing with small number of replicates. The software adopts a hierarchical empirical Bayes estimator for variance estimation. It pools information from all genes across the array and can significantly increase the sensitivity in gene selection when there are only a few replicates available.

5.3.1 How to use powexpress

In order to use *powexpress*, please follow the steps below.

(1) Prepare a data file that contains normalized expression data for all genes (probes/probesets). The file should follow EXPRESSION_DATA format (A.1.24).

(2) Prepare a parameter file that specifies the expression pattern one wish to screen for. The file format will be discussed below.

(3) Optionally, one can also prepare an annotation file that provides annotations for all genes. The file should follow EXPRESSION_ANNOTATION format (A.1.25). If this file is provided, the selected genes will be annotated based on the file. Notice that the ordering of annotations should match the ordering of probes/probesets in the data file, i.e. the first annotation line should correspond to the first probeset (the first data line) in the data file, etc.

(4) run *powexpress*. For example:

```
> powexpress -d data.txt -a datainfo.txt -c compinfo.txt -o gene_out
```

In *powexpress*,

“-d” specifies the data file that contains normalized expression data.

“-o” specifies the title for output file (section 5.3.3).

“-c” specifies the parameter file that specifies comparison criteria.

“-a” specifies the annotation file. One can set "-a NULL" in the command line if one does not have or does not want to provide an annotation file. In this case, the program will not link the selected probesets to gene annotations.

5.3.2 Parameter file

To select probesets that show patterns of interest, one need to provide a parameter file (say compinfo.txt) that specifies which pattern one wants to select, e.g. "mu1<mu2 and mu3>m4". A sample file is as below.

```
[Basic Info]
array number = 30
probeset number = 57753
group number = 10

[Group ID]
1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6 7 7 7 8 8 8 9 9 9 10 10 10

[Comparisons]
((1<2) & (2<3)) | ((4<5) & (5<6))

[Preprocessing Setup]
truncate lower bound = 2
take log2 before calculation (1:yes; 0:no) = 1

[Output Setup]
print top # of genes = 300

[Simulation Setup]
permutations for FDR calculation = 2
permutations within each sorting cycle = 1000

[Permutation Setup]
permutation group = 4
1 2 3
4 5 6
7 8
9 10

[Variance Setup]
variance group = 4
1 2 3
4 5 6
7 8
9 10
```

One can prepare a parameter file by modifying the sample file, but please do not change the format of the file, leave field labels such as "array number = " in their original form.

(A) In "Basic Info" section, one needs to provide general information about microarray experiment, including array number, probeset number and group number (i.e. number of experimental conditions).

(B) In "Group ID" section, one needs to label which group each array belongs to using an integer. For example, "1 1 1 2 2 2 3 3 3" means that the first array (2nd column in the data file) is from condition 1, the second array (3rd column in the data file) is from condition 1, ..., the fourth array (5th column in the datafile) is from condition 2, etc. Negative integers can be used if one wants to ignore a specific array. For example, "1 -1 1 2 -2 2 3 3 3" will ignore the second and fifth array in the analysis.

(C) In "Comparisons" section, one sets the selection criteria, i.e. expression patterns one wants to select. For example, the criteria can be specified as "((1>2) & (2>3)) | ((4<5) & (5<6))", in which case the program will select probesets whose expression level is ((higher in condition 1 than in condition 2) and (higher in condition 2 than in condition 3)) or ((lower in condition 4 than in condition 5) and (lower in condition 5 than in condition 6)). Currently, we only support the following operations:
< (less than)
> (greater than)
& (and)
| (or)

(D) In "Preprocessing Setup", one specifies how to truncate low expression values and whether log-transformation should be used before analysis. If "truncate lower bound = 2", all expression values in datafile which are less than 2 will be set equal to 2. If "take log2 before calculation (1:yes; 0:no) = 1", expression values will be log-transformed before performing test for each probeset.

(E) In "Output Setup", one specifies how many genes to report.

(F) In "Simulation Setup", one specifies how many iterations to run for FDR estimation and how many samples to draw for doing multiple sample comparisons.

If "permutations for FDR calculation = 10", the labels in "Group ID" will be permuted 10 times to compute a FDR. The permutation is done based on permutation groups set in "Permutation Setup" section. It will be done within each permutation group. For example, if there are four permutation groups as follows
1 2 3
4 5 6
7 8
9 10

then the permutation will be done within group 1,2 and 3, i.e. all arrays labeled by 1 or 2 or 3 will be permuted. Similary, arrays labeled by 4, 5 or 6 will be permuted, etc. No permutation will be done between two permutation groups, e.g. between (1,2,3) and (4,5,6). As a result, the FDR computed is a FDR for null hypothesis H0: "1=2=3, 4=5=6, 7=8, 9=10".

If "permutations within each sorting cycle = 1000", then 1000 Monte Carlo draws will be made to infer the probability that the selection criteria will be satisfied. This setup is only used for multiple sample comparisons such as "(1<2) & (2<3)". For two sample comparisons such as "1<2", a modified t-test will be used directly, no Monte Carlo simulation needs to be done.

(G) In "Permutation Setup", one can specify how to do permutations for FDR estimation. One needs to tell how many permutation groups there are. Then for each permutation group, one needs to specify which experimental conditions are included. For example:

permutation group = 2
1 2 3
4 5 6
means that there are 2 permutation groups. Arrays labled by 1, 2 or 3 will be mixed and permuted; arrays labled by 4, 5 and 6 will be mixed and permuted. The permutation will not bedone between arrays labled by 1 and 4, 1 and 5 etc. The null hypothesis is therefore "1=2=3, 4=5=6".

(H) In "Variance Group", one needs to specify which groups are assumed to have common variance. The variance shrinking will be based on the settings here. For example,
variance group = 2
1 2 3
4 5 6
means that we assume that group 1, 2 and 3 have the same within-condition variance, and group 4, 5 and 6 also have a common within-condition variance. But the within-condition variance for (1,2,3) and (4,5,6) can be different. The variance shrinking will be done within each variance group.

5.3.3 Output file

After specifying the title for output file, say “gene_out”, the program will generate four files as follows.

gene_out.txt -- top genes with annotations. The highest ranking (smallest test-statistics) probesets are reported. They are linked to annotations provided in the annotation file. The probesets are sorted from top to bottom according to the degree they match the selection criteria.

gene_out.ori -- the probeset level test-statistics for all probesets, the smaller the better. For two sample comparisons, this is the modified t-statistic. For multiple sample comparisons, this is 1-(posterior probability that the selection criteria are satisfied). This file will be used by "powexpressloc" for combining multiple probesets.

gene_out.pb -- probeset name

gene_out_ctr.txt -- a file that contains randomly chosen probes. This file may serve as a negative control for downstream analysis.

5.4 Get raw expression data for specified probesets

```
[Usage]
> powexpress_getspecificprobe -d [data file] -i [input file] -c [column
number of probeset id] -o [output file]
```

powexpress_getspecificprobe can be used to pick up specific probes and store their raw expression data to an independent file. The resulting file can be imported into dChip for visualization.

In order to use *powexpress_getspecificprobe*, one can follow the steps below.

(1) Prepare a data file in EXPRESSION_DATA format. The file stores the expression data for all probes/probesets in a microarray experiment (e.g., shhdata.txt).

(2) Prepare a file that specifies which probes need to be picked up (e.g., target.txt). The file is a tab-delimited text file, and one column should correspond to probe/probeset ID.

(3) Run *powexpress_getspecificprobe*. For example:

```
> powexpress_getspecificprobe -d shhdata.txt -i target.txt -c 0 -o
target_affy.txt
```

In *powexpress_getspecificprobe*,

"-d" specifies the data file that contains expression data.

"-i" specifies the input file that contains probe/probeset IDs of interest.

"-c" specifies which column in the input file are the probe/probeset IDs. The column is indexed from 0, i.e., 0 is equivalent to the first column, 1 is equivalent to the second column, etc.

"-o" specifies the output file.

5.5 Remove redundant probesets from the raw data file

```
[Usage]
> powexpress_getnrprobe -i [input file] -o [output file]
```

powexpress_getnrprobe can be used to get rid of redundant probes/probesets (i.e., probes/probesets that appear >=2 times) in a EXPRESSION_DATA file.

In order to use *powexpress_getnrprobe*, one can follow the steps below.

(1) Prepare a data file in EXPRESSION_DATA format. The file stores the original expression data.

(2) Run *powexpress_getnrprobe*. For example:

```
> powexpress_getnrprobe -i inputdata.txt -o data_nr.txt
```

In *powexpress_getnrprobe,*

"-i" specifies the input file that contains original expression data.

"-o" specifies the output file in which redundant probes/probesets are removed.

6. ChIP-chip Toolbox – ChIP-chip Peak Detection

6.1 Introduction

TileMap is a tool designed for tiling array analysis. It can be used to identify transcription factor binding regions from ChIP-chip tiling array data. This section will briefly introduce how one can use TileMap. More detailed information about TileMap can be found at http://biogibbs.stanford.edu/~jihk/TileMap/index.htm.

6.2 List of Functions

(1) *tilemap_importaffy* – import data from Affymetrix arrays.

(2) *tilemap_norm* – quantile normalization.

(3) *tilemap* – detect binding regions

(4) *tilemap_extract* – retrieve probes and summary statistics in user-specified regions. The retrieved data can be easily loaded into R, Matlab etc. for visualization.

(5) sample R/Matlab code tilemap_plot.R (or .m) for visualizing tilemap results.

6.3 ChIP-chip Analysis Step I – Import Raw Data

```
[Usage]
> tilemap_importaffy [importaffy_parameter_file]
```

tilemap_importaffy can be used to import data from Affymetrix arrays and preprocess the data (normalization & create local repeat filters). It converts *.CEL (version 3, ASCII file) and *.BPMAP files into the standard tilemap data format (see file format section).

In order to use *tilemap_importaffy*, one needs to have
(a) raw *.CEL (version 3, an ASCII file) files;
(b) *.BPMAP file (can be downloaded from affymetrix website);
(c) and a parameter file as below:

```
###############################
# TileMap Import Affymetrix   #
###############################

[Working                         directory]                         =
C:\Projects\research_harvard\affy_project\analysis\tiling_paper\
[BPMAP file] = P1_CHIP_A.Anti-Sense.hs.NCBIv33.sary.bpmap
[Export file] = cMycA_tile.txt

###############################
# Arrays                  #
###############################
[Array number] = 18
[Arrays]
IP_5_3A.CEL   Jurkat_anti-cMyc_A_1_1
IP_5_4A.CEL   Jurkat_anti-cMyc_A_1_2
IP_5_5A.CEL   Jurkat_anti-cMyc_A_1_3
IP_1_3A.CEL   Jurkat_anti-GST_A_1_1
IP_1_4A.CEL   Jurkat_anti-GST_A_1_2
IP_1_5A.CEL   Jurkat_anti-GST_A_1_3
IP_7_3A.CEL   Jurkat_Input_A_1_1
IP_7_4A.CEL   Jurkat_Input_A_1_2
IP_7_5A.CEL   Jurkat_Input_A_1_3
IP_6_1A.CEL   Jurkat_anti-cMyc_A_2_1
IP_6_2A.CEL   Jurkat_anti-cMyc_A_2_2
IP_6_3A.CEL   Jurkat_anti-cMyc_A_2_3
IP_2_1A.CEL   Jurkat_anti-GST_A_2_1
IP_2_4A.CEL   Jurkat_anti-GST_A_2_2
IP_2_3A.CEL   Jurkat_anti-GST_A_2_3
IP_8_1A.CEL   Jurkat_Input_A_2_1
```

```
IP_8_2A.CEL   Jurkat_Input_A_2_2
IP_8_3A.CEL   Jurkat_Input_A_2_3

################################
# Normalization             #
################################
[Apply normalization before computing intensity] (1:yes; 0:no) = 1
[Truncation lower bound before normalization] = -100000000.0
[Take log2 transformation before normalization] (1:yes; 0:no) = 0

################################
# Method to Compute Intensity #
################################
[How to compute intensity] (0: PM only, 1: PM-MM) = 0
[Truncation lower bound after intensity computation] = -100000000.0
[Take log2 transformation after intensity computation] (1:yes; 0:no) =
0
```

Below are the meanings for each item in the parameter file.

[Working directory]
The directory that contains *.CEL and *.BPMAP files. All the results generated by TileMap will be exported to this directory.

[BPMAP file]
The name of the *.BPMAP file. This file should be placed in the working directory. It will be used to sort the probes according to their genomic location and to generate a local repeat filter.

[Export file]
Please specify a file to save the converted data. The raw *.CEL data will be exported into [working directory]\[export file] in the standard tilemap data format.

[Array number]
Number of arrays.

[Arrays]
Each line below [Arrays] represent an array. Each line contains two columns, separated by a tab, the first column gives the name of the *.CEL file, and the second column gives the name of the array (provided by users to specify e.g. experimental conditions ...). For example:

IP_5_3A.CEL Jurkat_anti-cMyc_A_1_1
IP_5_4A.CEL Jurkat_anti-cMyc_A_1_2

IP_5_5A.CEL Jurkat_anti-cMyc_A_1_3
IP_1_3A.CEL Jurkat_anti-GST_A_1_1
IP_1_4A.CEL Jurkat_anti-GST_A_1_2
IP_1_5A.CEL Jurkat_anti-GST_A_1_3

The number of *.CEL files should match the [Array number].

[Apply normalization before computing intensity]
Whether or not you want to do normalization before computing probe intensities.

[Truncation lower bound before normalization]
If you choose to do normalization, you need to specify how to truncate low expression values. All values < [truncation lower bound] will be set to [truncation lower bound] before normalization.

[Take log2 transformation before normalization]
Whether or not you wish to take log2 transformation before normalization. If you choose yes, the truncated values will be log-transformed, and the normalization will be applied to the transformed values. If you choose no, the normalization will be applied to the un-log-transformed values, and you can choose to do log-transformation later.

[How to compute intensity]
You can choose to use normalized PM values as the probe intensity; or you can choose to use PM-MM as the intensity.

[Truncation lower bound after intensity computation]
After you compute PM only or PM-MM intensities, how would you truncate low intensities. All intensities < [truncation lower bound] will be set to [truncation lower bound]. If you have already taken log-transformation before, you may need to set a small number here such as -10000000000.0.

[Take log2 transformation after intensity computation]
Whether or not you wish to take log2 transformation after you get intensities. If you have already carried out log-transformation before normalization, you should choose "no" here.

6.4 ChIP-chip Analysis Step II – Normalization

```
[Usage]
> tilemap_norm [norm_parameter_file]
```

tilemap_norm can be used to do quantile normalization (Bolstad et al., 2003). If users want to analyze non-affymetrix data, they can provide their own data in the standard tilemap data format, and use tilemap_norm to do normalization. Users do not need to use tilemap_norm if tilemap_importaffy was used to import raw data. tilemap_importaffy already provides options to do normalization.

In order to run tilemap_norm, one also needs to have
(a) A raw data file in TILEMAP_DATA format.
(b) a parameter file as below:

```
###############################
# TileMap Normalization      #
###############################

[Working directory] = .
[Raw Data file] = sample2_raw.txt
[Export file] = sample2_norm.txt
[Array number] = 18
[Truncation lower bound before normalization] = 1.0
[Take log2 transformation before normalization] (1:yes; 0:no) = 1
```

Below are the meanings of parameter settings.
[Working directory]
The directory that contains the raw data file. All the results generated by TileMap will be exported to this directory.

[Raw Data file]
The name of the raw data file. It should be placed in the working directory and should be in standard tilemap data format.

[Export file]
The name of the file where normalized data will be saved. This file will be generated in the working directory.

[Array number]
Number of arrays.

[Truncation lower bound before normalization]
You need to specify how to truncate low expression values. All values < [truncation

lower bound] will be set to [truncation lower bound] before normalization.

[Take log2 transformation before normalization]
Whether or not you wish to take log2 transformation before normalization. If you choose yes, the truncated values will be log-transformed, and the normalization will be applied to the transformed values. If you choose no, the normalization will be applied to the un-log-transformed values, and you can choose to do log-transformation later in tilemap.

6.5 ChIP-chip Analysis Step III – Peak Detection

```
[Usage]
> tilemap [tilemap_parameter_file]
```

tilemap is the central part of TileMap. It (i) computes probe-level test-statistics according to the transcriptional or protein binding patterns specified by users; (ii) filters local repeats; and (iii) infers if a region is of interest or not by applying HMM or Moving Window Average (MA). The output of tilemap includes *.sum files which provide summary statistics for each probe, a *.bed file which reports regions of interest, and a *.reg file which sorts the reported regions from high to low significance level. *.bed file can be uploaded directly to UCSC genome browser to visualize the reported regions.

In order to run tilemap, one needs to have

(a) A normalized data file in TILEMAP_DATA format;

(b) A *.cmpinfo file in TILEMAP_COMPINFO format to specify which pattern one wish select;

(c) A parameter file as below:

```
##############################
# TileMap Parameter Settings #
##############################

##############################
# Step O: Working Directory  #
##############################
O.1-[Working                       directory]                        =
C:\Projects\research_harvard\affy_project\analysis\tiling_paper\
O.2-[Project Title] = cMycA_tile

##############################
# Step I: Probe Summary     #
##############################
I.1-[Compute probe level test-statistics?] (1:yes; 0:no) = 0
I.2-[Raw data file] = cMycA_tile_I_f_pb.sum
I.3-[Range of test-statistics] (0: default; 1: [0,1], 2: (-inf, +inf))
= 1
I.4-[Zero cut] = 1e-4

##############################
# Step II: Repeat Filtering  #
##############################
II.1-[Apply local repeat filter?] (0:No; 1:Yes) = 0
II.2-[*.refmask                       file]                        =
P1_CHIP_A.Anti-Sense.hs.NCBIv33.sary.bpmap.refmask

##############################
# Step III: Region Summary   #
##############################
III.1-[Combine neighboring probes?] (0:No; 1:Yes) = 1
III.2-[Method to combine neighboring probes] (0:HMM, 1:MA) = 1

##############################
# Step IV: HMM             #
##############################
IV.1-[Posterior probability >] = 0.5
IV.2-[Maximal gap allowed] = 1000
IV.3-[Method to set HMM parameters] (0:UMS, 1:Set by users) = 0

IV.4-[Provide your own selection statistics?] (0: No, use default; 1: Yes)
= 0
IV.5-[If Yes to IV.4, selection statistics file] = cMycA_tile_I_f.pbsum
IV.6-[G0 Selection Criteria, p%] = 0.01
IV.7-[G1 Selection Criteria, q%] = 0.05
IV.8-[Selection Offset] = 1
IV.9-[Grid Size] = 1000
IV.10-[Expected hybridization length] = 28

IV.11-[Path to transition probability matrix] = cMycA_tile_transp.txt
IV.12-[Path to emission probability matrix] = cMycA_tile_emissp.txt

##############################
# Step V: Moving Average    #
##############################
V.1-[Local FDR <] = 0.5
V.2-[Maximal gap allowed] = 500
V.3-[W] = 5
V.4-[Method to compute local FDR] (0:UMS; 1:Permutation Test) = 0

V.5-[Provide your own selection statistics?] (0: No, use default; 1: Yes)
= 0
V.6-[If Yes to IV.4, selection statistics file] = cMycA_tile_I_f.pbsum
V.7-[G0 Selection Criteria, p%] = 0.01
V.8-[G1 Selection Criteria, q%] = 0.05
V.9-[Selection Offset] = 6

V.10-[Grid Size] = 1000
```

Here are the meanings for tilemap parameter settings.

O.1-[Working directory]
The directory that contains raw data files. All the results generated by TileMap will be exported to this directory.

O.2-[Project Title]
A title of the project. This title will be used to generate names of output files.

I.1-[Compute probe level test-statistics?]
Specify whether or not tilemap should compute the probe level test-statistics. If one only has normalized raw data, one should choose "yes". If one has already pre-computed probe level test-statistics and only wants to apply HMM or MA to do region level inference, one can choose "no".

I.2-[Raw data file]:
If you choose "yes" in I.1, you need to prepare two files in the working directory:
(i) A raw data file which contains the normalized probe intensities. This file should be in standard tilemap data format, and should be placed in the working directory. Give its name in I.2.
(ii) You also need to prepare a *.cmpinfo file named {Project Title}.cmpinfo in the working directory, which specifies the hybridization pattern you wish to select. However, you DON'T need to provide its name in I.2.
If you choose "no" in I.1, please prepare a file that contains precomputed probe level test-statistics. This file should be in *_pb.sum format (see "Output File Format") and should be placed in the working directory. Provide its name in I.2. It will be used as the input for HMM and MA. In this case, the probe level computation embedded in TileMap will be skipped.
NOTICE: in tilemap, small values of probe level test-statistics correspond to patterns of interest. When you provide your own probe level test-statistics, you may need to transform them somehow to follow this convention.

I.3-[Range of test-statistics]
Specify the range of probe level test-statistics.
If you choose "yes" in I.1, you can set I.3 to 0 (default). Tilemap will compute probe level test statistics and determine the range automatically. For two sample comparisons, the probe level test-statistic is an improved t-statistic, the range will be (-inf, +inf). For multiple sample comparisons, the probe level test-statistic is a posterior probability, the range will be [0,1].
If you choose "no" in I.1 and provide your own probe level test-statistics, then you should set I.3 either to 1 {[0,1]} or 2 {(-inf, +inf)} depending on whether the statistics you provided in I.2 fall within [0,1] (e.g. posterior probability) or (-inf, +inf) (e.g. t-statistics).

[0,1] statistics will be transformed by log[t/(1-t)] before applying MA, and the MA statistics will be transformed back by exp(u)/[exp(u)+1] before applying UMS to estimate local FDR. (-inf, +inf) statistics will be transformed by exp(t)/[exp(t)+1] before applying HMM.

I.4-[Zero cut]
To avoid logit(0), please specify a zero cut in I.4. [0,1] test-statistics will be set to max(zero_cut/2, min(t, 1-zero_cut/2)) before taking logit transformation. E.g. If [Monte Carlo draws for posterior prob.] = 10000 in *.cmpinfo file, you can set zero_cut = 0.0001.

II.1-[Apply local repeat filter?]
Whether or not you want to mask local repeats. Some probes occur more than once in a region, such local repeats may result in noise due to cross-hybridizations. You may wish to exclude these probes from analysis. If so, you need to apply the filter. If the data you provided have already been repeat-masked, you can choose "no" to skip this step.

II.2-[*.refmask file]
If you choose "yes" in II.1, please prepare a *.refmask file (see "Output File Format") which provides non-redundant probes and counts how many times each probe occur in a local region. You need to provide its name in II.2. This file will be used as a reference for masking local repeats.
Hint: using tilemap_importaffy to load affymetrix data from *.CEL and *.BPMAP will automatically create a *.refmask file.
If you choose "no" in II.1, set II.2 to NULL. Local repeat filtering will be skipped.

III.1-[Combine neighboring probes?]
Whether or not you want to apply HMM or MA to do region inference. If you choose no, tilemap will skip HMM and MA. If you choose yes, tilemap will combine neighboring probes to infer whether a region is of interest or not.

III.2-[Method to combine neighboring probes]
Choose which method should be used to do region summary.
If you choose "Yes" in III.1 and "HMM" in III.2, please fill out Step IV and leave Step V to its default values.
If you choose "Yes" in III.1 and "MA" in III.2, please fill out Step V and leave Step IV to its default values.
If you choose "No" in III.1, you can set III.2 arbitrarily to 0 or 1 and leave both Step IV and Step V to their default values. Region summary will then be skipped.

IV.1-[Posterior probability >]
Posteriror probability cutoff to call regions of interest in HMM.

IV.2-[Maximal gap allowed]
d0 in HMM. If the distance between the neighboring probes i and i+1, d(i,i+1), is no greater than d0, tilemap will use the HMM transition probability matrix to compute likelihood. If d(i,i+1) > d0, tilemap will restart a new HMM from i+1. (refer to Ji&Wong, 2005 for details).

IV.3-[Method to set HMM parameters]
You can choose to use UMS embedded in tilemap to get HMM parameters or to provide your own HMM parameters.
If you choose "UMS" in IV.3, please set UMS parameters in IV.4 - IV.10. Otherwise leave them to be default values.
If you choose "Set by users" in IV.3, please provide your own transition, emission probability matrices in _transp.txt and _emissp.txt format. You should place these two files in the working directory and provide their names in IV.9 and IV.10. Otherwise set IV.9 and IV.10 to be NULL.

IV.4-[Provide your own selection statistics?]
If you choose to use UMS to get HMM parameters, you have the option to provide your own selection statistics. If you do not provide your own selection statistics, tilemap will use the probe level test-statistics as the default selection statistics.

IV.5-[If Yes to IV.4, selection statistics file]
If you choose to provide your own selection statistics, please prepare the statistics in a *_pb.sum file and provide its name in IV.5. The file should be in working directory.

IV.6-[G0 Selection Criteria, p%]
Set t(p) in UMS. If a probe has a selection statistic > t(p), its downstream probe will be used to construct g0.

IV.7-[G1 Selection Criteria, q%]
Set t(q) in UMS. If a probe has a selection statistic <= t(q), its downstream probe will be used to construct g1.

IV.8-[Selection Offset]
If probe i has a selection statistic > t(p), probe (i+selection_offset) will be used to construct g0. Similar for g1.

IV.9-[Grid Size]
How many intervals should [0,1] be divided into. For example, if grid size = 1000, [0,1] will be divided into 0.001, 0.002, ..., 1.000. g0 and g1 will be estimated by empirical distributions on this grid. The choice of grid size should consider the number of probes available. On average, it would be better to have a few hundred probes in each interval.

IV.10-[Expected hybridization length]
The number of probes contained in a typical hybridization region. For example, in ChIP-Chip experiment, if IP fragment length = 1000bp, probe density= 1 probe / 35 bp. Then one would expect to observe 28 probes on average in a binding region, and one can set expected hybridization length = 28.

IV.11-[Path to transition probability matrix]
If you choose "Set by users" in IV.3, please prepare a transition probability matrix in working directory and in _transp.txt format (see "Output File Format"). Provide its name here.

IV.12-[Path to emission probability matrix]
If you choose "Set by users" in IV.3, please prepare a emission probability matrix in working directory and in _emissp.txt format (see "Output File Format"). Provide its name here.

V.1-[Local FDR <]
Local false discovery rate cutoff to call regions of interest in MA.

V.2-[Maximal gap allowed]
Two signifcant probes, if their distance <= [maximal gap allowed], will be treated as a single region. For example, in ChIP-Chip experiment, if IP fragment length = 1000bp, one can set maximal gap allowed = 500, half of the IP fragment length.

V.3-[W]
The half window size. The moving average will be taken over a 2*W+1 window, i.e. each window will contain 2*W+1 probes.

V.4-[Method to compute local FDR]
You can choose to use UMS or permutation test to compute local FDR.
If you choose "UMS" in V.4, please set UMS parameters in V.5 - V.10. If you choose "Permutation Test" in V.4, please set grid size in V.10, and then go back to {Project Title}.cmpinfo file and fill out its "Permutation Setup" section. There you will set the way to do permutations and number of permutations you want to do.
Hint: depending on the size of the data, permutation test could be very slow.

V.5-[Provide your own selection statistics?]
If you choose to use UMS to get local FDR, you have the option to provide your own selection statistics in UMS. If you choose not to provide your own selection statistics, tilemap will use probe level test-statistics as the selection statistics.

V.6-[If Yes to IV.4, selection statistics file]
If you choose to provide your own selection statistics, please prepare the statistics in a *_pb.sum file and provide its name here. The file should be in working directory.

V.7-[G0 Selection Criteria, p%]
Set t(p) in UMS. If a probe has a selection statistic > t(p), its downstream probe will be used to construct g0.

V.8-[G1 Selection Criteria, q%]
Set t(q) in UMS. If a probe has a selection statistic <= t(q), its downstream probe will be used to construct g1.

V.9-[Selection Offset]
If probe i has a selection statistic > t(p), probe (i+selection_offset) will be used to construct g0. Similar for g1. Usually, selection offset = W+1 in MA.

V.10-[Grid Size]
How many intervals should [0,1] be divided into. For example, if grid size = 1000, [0,1] will be divided into 0.001, 0.002, ..., 1.000. g0 and g1 will be estimated by empirical distributions on this grid. The choice of grid size should consider the number of probes available. On average, it would be better to have a few hundred probes in each interval.

6.6 ChIP-chip Analysis Step IV – Get Raw Intensity Data for Specified Genomic Regions

```
[Usage]
> tilemap_extract [extract_parameter_file]
```

tilemap_extract can be used to retrieve probes and their summary statistics in user-specified regions. The retrieved data will be saved in tab-delimited ASCII files. These files can be easily loaded into R, Matlab etc. for visualization.

In order to run tilemap_extract, one needs to have

(a) summary statistics computed by tilemap;
(b) a parameter file as below:

```
[Working directory] = .
[Project Title] = sample1
[Probe Level Summary] = sample1_f_pb.sum
[Raw Data] = sample1_rawdata.txt
[Regions]
chr21  14676034   14678449   target 651 +
chr21  17421450   17423637   target 651 +
chr21  18111757   18113819   target 651 +
chr21  26027537   26031330   target 651 +
```

The meanings of each item in the parameter file are explained below.

[Working directory]
The directory that contains raw data file and all the tilemap-generated files. All the new files generated by this command will be exported to this directory.

[Project Title]
The title of the project. The program will automatically extract data from [Project Title]_f_pb.sum, [Project Title]_hmm.sum, [Project Title]_ma.sum and the raw data file if available.

[Probe Level Summary]
You are required to provide a probe level summary file in _pb.sum format. The program will decide which probes to retrieve based on this file.

[Raw Data]
You can provide the name of the raw data file. This file should be in working directory. If provided, raw data will be extracted. If you set NULL here, the program will only get test-statistics for target probes. No raw data will be extracted.

[Regions]
Each line below [Regions] represent a region you wish to extract. Each line contains at least three columns, tab-delimited.

col1: chromosome name
col2: start coordinate in the chromosome
col3: end coordinate in the chromosome
other columns: defined by users themselves.

e.g.

```
chr21 14676034 14678449 target 651 +
chr21 17421450 17423637 target 651 +
chr21 18111757 18113819 target 651 +
chr21 26027537 26031330 target 651 +
```

for each region, the program will generate a file named [Project title]_[chromosome]_[start]_[end].txt in the working directory. The file will contain all the probes in the specified region, their coordinates and summary statistics.

6.7 ChIP-chip Analysis Step V – Visualization

Since TileMap does not provide an integrated GUI at current stage, we provide tilemap_extract and R/Matlab sample code to facilitate the visual checking of the data and tilemap results. We plan to delevop an integrated visualization system in future which will integrate TileMap with downstream analysis programs. The sample code can be downloaded from http://biogibbs.stanford.edu/~jihk/TileMap/TileMap/tilemap_plot.R and http://biogibbs.stanford.edu/~jihk/TileMap/TileMap/tilemap_plot.m.

In order to visualize the data, run these programs as below.

(1) In MatLab:

```
> tilemap_plot('[file name]')
```

e.g.

```
> tilemap_plot('cMycA_tile_chr21_14676034_14678449.txt')
```

(2) In R:

First, in tilemap_plot.R find a line started with "datapath <- ", and edit the line to provide a filename that specifies the data for plotting, e.g.

```
datapath <- "cMycA_tile_chr21_14677034_14677449.txt"
```

Then run tilemap_plot.R

(3) Users can modify Matlab and R codes to meet their own needs.

6.8 ChIP-chip Analysis – TileMap output files

```
/* ------------------------ */
/* Output File Format      */
/* ------------------------ */

##################################
# *.refmask                      #
##################################
```

This is a tab-delimited file used for sorting probes based on their genomic coordinates and for filtering local repeats.

col1: chromosome name
col2: coordinate in the chromosome
col3: how many probes in the array are mapped (without any mismatches) to the position specified by col1 and col2.
col4: within a local window (2000 bp as the tilemap default) centered at the col1-col2, how many genomic loci have the same probe sequence as the sequence specified by col1-col2. If the number is >1, the probe in question will be treated as a local repeat and will be filtered out later.
col5: probe sequence

```
##################################
# *_pb.sum                   #
##################################
```

This is a tab-delimited file to record probe level test-statistics.

col1: chromosome name
col2: coordinates in the chromosome
col3: probe-level test-statistics. The statistics are transformed such that the smaller the statistics, the more significant.

```
##################################
# *_hmm.sum                  #
##################################
```

This is a tab-delimited file to record posterior probability generated by HMM.

col1: chromosome name
col2: coordinates in the chromosome
col3: posterior probability that a probe is in a region of interest. The larger the posterior probability, the more significant a probe is.

```
###################################
# *_ma.sum                        #
###################################
```
This is a tab-delimited file to record MA summaries.

col1: chromosome name
col2: coordinates in the chromosome
col3: Moving average (MA) statistics
col3: Local false discovery rate that a probe is in a region of interest. The smaller the local FDR, the better.

```
###################################
# *.bed                          #
###################################
```
This is a UCSC *.bed file to report significant regions. Regions are sorted according to their genomic locations.

col1: chromosome name
col2: region start
col3: region end
col4: no meaning
col5: 1000*[hmm posterior probability] or 1000*(1-lfdr of MA)
col6: always +

```
###################################
# *.reg                          #
###################################
```
This is a tab-delimited file to report significant regions. Regions are ranked according to their significance levels.

col1: chromosome name
col2: start coordinate
col3: end coordinate
col4: the line # of the starting probes (to help locate the probe in *_pb.sum, *_hmm.sum and *_ma.sum files)
col5: the line # of the ending probe
col6: maximum posterior probability or minimum local FDR of all the probes in the region
col7: mean posterior probability or mean local FDR of the regions. If the region is formed by merging two discrete regions that are separated by less than [maximal gap], then the mean is obtained as follows: first, compute two means for the two discrete regions separately; then, take the

minimum of the two means and use it as the mean here.

```
##################################
# _transp.txt                    #
##################################
```

HMM transition probability matrix, in the format:

```
1-a0 a0
a1 1-a1
```

```
##################################
# _emissp.txt                    #
##################################
```

HMM emission probability matrix, in the format:

```
interval(1) interval(2) interval(3) ... interval(n)
f0(1) f0(2) f0(3) ... f0(n)
f1(1) f1(2) f1(3) ... f1(n)
```

A probe level test-statistic, t, if interval(i-1)<t<=interval(i), then f0(t)=f0(i) (the likelihood for H=0); and f1(t)=f1(i) (the likelihood for H=1).

NOTICE: interval(i) should equally divide [0,1], i.e. interval(i+1)-interval(i) = interval(i)-interval(i-1). interval(n) is always 1.0. Although not explicitly defined in the file, interval(0)=0.

```
##################################
# file exported by               #
# tilemap_extract                #
##################################
```

This is a tab-delimited file.

col1: probe coordinate in chromosome
col2: probe level test-statistics
col3: HMM posterior probability
col4: MA statistics
col5: local FDR for MA
col6 and after: raw data

7. DE NOVO MOTIF DISCOVERY TOOLBOX – DE NOVO MOTIF DISCOVERY

7.1 Introduction

When the binding motif of a transcription factor is unknown, one may want to figure out what it is. Often, such a motif can be recovered from a set of sequences in which the motif binding sites are assumed to be enriched. This section will introduce functions provided by CisGenome for discovering such unknown motifs.

7.2 List of Functions

(1) *flexmodule_motif* – *De novo* motif discovery based on a collapsed Gibbs Motif Sampler.

(2) *flexmodule_tnum* – *De novo* motif discovery that favors TFBS physically clustered together.

(3) *flexmodule* – *De novo* motif/module discovery where users can specify module structures (this function will be discussed elsewhere).

7.3 De novo Motif Discovery I – A Collapsed Gibbs Motif Sampler

```
[Usage]
Flexmodule_motif [parameter file]
```

flexmodule_motif can be used to search for enriched sequence motifs in a set of DNA sequences. The search is based on a collapsed Gibbs Motif Sampler (Lawrence et al., 1993; Liu, 1994; Liu et al., 1995). Multiple motifs can be searched simultaneously.

7.3.1 How to use *flexmodule_motif*

flexmodule_motif can be used as follows.

(1) Prepare a file that contains all input sequences in FASTA format. Sequences are treated as soft-masked. In other words, small letters "a", "c", "g" and "t", as well as "N" will be ignored in the motif discovery. Only capital letters "A", "C", "G" and "T" will be involved in the *de novo* motif discovery.

(2) Prepare a main parameter file (e.g., `flexmodule_motif_arg.txt`) and a prior motif abundance file (e.g., `priorabundance.txt`) as will be discussed below.

(3) Run *flexmodule_motif*. For example:

```
> flexmodule_motif flexmodule_motif_arg.txt
```

(4) Check results in a file named *_p.txt

In order to run *flexmodule_motif*, one needs to prepare parameter files to specify the number of motifs, initial motif lengths, prior abundance for each motif, and number of iterations for Markov Chain Monte Carlo (MCMC). The information is usually provided in two parameter files, a *main parameter file* and a *prior motif abundance file*. Optionally, one can also incorporate his/her prior knowledge about motif PWMs into motif discovery. If so, one needs to prepare additional files in MOTIF_MAT format to specify the prior. Next, we discuss parameter files in more detail.

7.3.2 Main Parameter File

Below is a sample main parameter file *flexmodule_motif_arg.txt*:

```
[Working Directory] = .
[FASTA Sequence] = Gli.fa
[Output File] = Gli_motif
[Use *.cs?] (0:No; 1:Yes) = 0
[CS Prefix] = NULL
[CS Likelihood f] = cslike.txt
[Motif Number K] = 3
[Mean Motif Length Lamda] = 12
```

```
[Maximal Motif Length Allowed] = 30
[Init Motif Length L] = 12,12,12
[Init Motif Matrix]
NULL
GGGGTGGGG.txt
NULL
[Init Motif Abundance] = priorabundance.txt
[Init Module Size D] = 2.0
[Module Length] = 150
[Sample Module Length?] (0:No; 1:Yes) = 0
[Order of Background Markov Chain] = 3
[Use Fitted Background?] (0:No; 1:Yes) = 0
[Fitted Background Prefix] = NULL
[MCMC Iteration] = 5000
```

In this file,

"[Working Directory] =" specifies the directory that stores input and output files. Both the input FASTA file and the prior motif abundance file should be saved in this directory. Any files that specify prior motif PWMs should also be stored here. This is also the directory where motif discovery results will be saved. One can use a dot (i.e., ".") to represent current directory.

"[FASTA Sequence] =" specifies the file that contains input sequences. The file should be stored in the working directory.

"[Output File] =" specifies a title for the output. Various output files will be generated using this title as part of their file names.

"[Use *.cs?] (0:No; 1:Yes) = 0": no use here, set as it is.

"[CS Prefix] = NULL": no use here, set as it is.

"[CS Likelihood f] = cslike.txt": no use here, set as it is.

"[Motif Number K] =" specifies the number of motifs. K different motifs will be searched simultaneously by the motif sampler.

"[Mean Motif Length Lamda] =" specifies the mean motif length. A priori, the motif length is modeled as a Poisson distribution with the mean equal to the number specified here. The distribution is truncated at , i.e., the minimum motif length is 8.

"[Maximal Motif Length Allowed] =" specifies the maximum motif length.

"[Init Motif Length L] =" specifies the initial motif length. The search will starts from the seeds that have the specified length. After certain iterations, the program will start to automatically adjust motif lengths. In the example above, "12,12,12" means that the initial length of the three motifs are all 12.

"[Init Motif Matrix]" specifies the prior for motifs. If there are K motifs, this line should be followed by K lines corresponding to K motifs. If a line specifies a prior as "NULL", a flat prior will be automatically used for the corresponding motif. If one wish to use a user-specified prior for a motif, then one should first prepare a file in MOTIF_MAT format that contains prior counts, next one should write down the file name here (e.g., "GGGGTGGGG.txt"). Please make sure that the prior PWMs are saved in the working directory.

"[Init Motif Abundance] =" specifies the name of the prior motif abundance file. Please make sure that the file is saved in the working directory.

"[Init Module Size D] = 2.0": no use here, set as it is.

"[Module Length] = 150": no use here, set as it is.

"[Sample Module Length?] (0:No; 1:Yes) = 0": no use here, set as it is.

"[Order of Background Markov Chain] =" specifies the order of background Markov model. This model will be used to describe non-motif sequences.

"[Use Fitted Background?] (0:No; 1:Yes) = 0": no use here, set as it is.

"[Fitted Background Prefix] = NULL": no use here, set as it is.

"[MCMC Iteration] =" specifies the number of iterations of MCMC.

7.3.3 Prior Motif Abundance File

Below is a sample prior motif abundance file *priorabundance.txt*:

```
1000 0
2 2
2 2
2 2
```

If there are K motifs, then sequences can be viewed as a mixture of a background model and K different motif models. Each motif can appear on forward or backward strands, therefore the TFBS can be grouped into 2K classes, i.e., motif 1 sites on + strand, motif 1 site on – strand, …, motif K sites on + strand, and motif K sites on – strand. The prior motif abundance file is a text file that specifies the relative abundance level for each of the 2K classes as well as background base pairs. The prior abundance is specified as positive numbers in a (K+1)x2 matrix. The first column of the matrix specifies the prior count for + strand, and the second column specifies the prior count for – strand. The first row is the prior count for background model, and each of the remaining lines specifies prior counts for a motif model. Notice that in the first line, the second number should always be 0. This matrix will be used as parameters of a Dirichlet distribution to characterize the prior abundance of each motif.

7.3.4 Output File

The motif discovery results will be reported in a file named [Output File]_p.txt. In the output file, motifs will be reported one by one in the following format:

```
****** Motif0 ******
Motif Score: 1.819343
Motif Matrix:
 5.0000000e-01  1.5000000e+00  1.9500000e+01  1.5000000e+00
 1.3500000e+01  5.5000000e+00  3.5000000e+00  5.0000000e-01
 3.5000000e+00  1.7500000e+01  5.0000000e-01  1.5000000e+00
 5.0000000e-01  2.1500000e+01  5.0000000e-01  5.0000000e-01
 1.9500000e+01  5.0000000e-01  5.0000000e-01  2.5000000e+00
 5.0000000e-01  2.1500000e+01  5.0000000e-01  5.0000000e-01
 4.5000000e+00  1.6500000e+01  1.5000000e+00  5.0000000e-01
 2.5000000e+00  1.9500000e+01  5.0000000e-01  5.0000000e-01
 1.9500000e+01  1.5000000e+00  1.5000000e+00  5.0000000e-01
 9.5000000e+00  6.5000000e+00  6.5000000e+00  5.0000000e-01
 2.5000000e+00  2.5000000e+00  1.4500000e+01  3.5000000e+00
 2.5000000e+00  6.5000000e+00  6.5000000e+00  7.5000000e+00
 1.0500000e+01  7.5000000e+00  2.5000000e+00  2.5000000e+00
 5.0000000e-01  1.5000000e+00  2.0500000e+01  5.0000000e-01
 7.5000000e+00  9.5000000e+00  3.5000000e+00  2.5000000e+00
 5.5000000e+00  7.5000000e+00  8.5000000e+00  1.5000000e+00

Consensus:
GACCACCCAAGTAGCG

Motif Sites:
0       211     226     +       GCTCACCAAAGTAGAG        0.170400
0       228     243     +       GACCACCCAGGTAGGC        0.842400
1       754     769     -       GACCACCCAGGACGCG        0.956000
3       426     441     -       TGCCACCCAATTAGCC        0.204000
3       630     645     +       GACCACCCAAGGTGAT        0.771200
4       227     242     -       GACCACCCACGCCGAG        0.988800
5       566     581     -       GACCACCCAGCGCGCG        0.796800
6       118     133     -       GACCACCCAGGAGCAG        0.484800
7       415     430     -       GACCACCCAAGCAGCA        0.974400
9       580     595     +       GACCACACAGGGCGTC        0.410400
10      313     328     +       GCCCACCCAAGTCGCC        0.969600
11      264     279     -       GCCCACACACAGCGCC        0.160800
12      356     371     -       GAACACCCAAGTAGAA        0.844800
```

Here, “Motif Matrix” gives the PWM that counts A, C, G and T respectively at each position (this is similar to MOTIF_MAT format).

“Consensus” is the consensus binding pattern derived from the PWM.

“Motif Sites” are predicted TFBS of the motif, and these sites are used to construct the reported matrix.

For each PWM, a “Motif Score” is computed as follows. Suppose n_{ij} is the count for base *j* (=A, C, G, T) at the *i*th position. $n_i = n_{iA}+n_{iC}+n_{iG}+n_{iT}$, $p_{ij}=n_{ij}/n_i$, q_j is the occurrence frequency of base *j* in the background (usually derived from all the input sequences). *W* is the length of the motif. The motif score *S* is defined as:

$$S = \frac{\sum_{i=1}^{L} \log(n_i) \sum_{j=A}^{T} p_{ij} \log(p_{ij}/q_j)}{W}$$

7.4 De novo Motif Discovery II – A Gibbs Motif Sampler that favors TFBS physically clustered together

```
[Usage]
Flexmodule_tnum [parameter file]
```

flexmodule_tum can be used to search for enriched sequence motifs in a set of DNA sequences. The search will be biased to regions where multiple TFBS are clustered together.

flexmodule_tum can be used in a similar manner as *flexmodule_motif*, but users need to specify a module length L. For each putative TFBS, a 2L base pair flanking window centered at the TFBS will be formed. The total number of putative TFBS within the window will be counted. The higher the number, the more likely the center site will be sampled as a real TFBS. Details about the model will be presented elsewhere. Conceptually, this is similar to a Module sampler.

To run *flexmodule_tum*, follow the same steps as *flexmodule_motif*, but make the following modifications in the main parameter file:

(1) Use "[Module Length]=" to specify the module length L.

(2) Use "[Init Module Size D] = 2.0" to specify what level of TFBS co-localization is required to make a TFBS functional. For example, if it is believed that a module of length 2L=2*150=300bp should contain >=3 sites in order to be considered as a high quality module, then one can set D=3-1=2.

Below is a sample main argument file for *flexmodule_tnum*:

```
[Working Directory] = .
[FASTA Sequence] = Gli.fa
[Output File] = Gli_motif
[Use *.cs?] (0:No; 1:Yes) = 0
[CS Prefix] = NULL
[CS Likelihood f] = cslike.txt
[Motif Number K] = 3
[Mean Motif Length Lamda] = 12
[Maximal Motif Length Allowed] = 30
[Init Motif Length L] = 12,12,12
[Init Motif Matrix]
NULL
GGGGTGGGG.txt
NULL
[Init Motif Abundance] = priorabundance.txt
[Init Module Size D] = 2.0
[Module Length] = 150
[Sample Module Length?] (0:No; 1:Yes) = 0
[Order of Background Markov Chain] = 3
[Use Fitted Background?] (0:No; 1:Yes) = 0
[Fitted Background Prefix] = NULL
```

```
[MCMC Iteration] = 5000
```

After the main parameter file and the prior motif abundance file are prepared, run the function as follows:

```
> flexmodule_tnum flexmodule_motif_arg.txt
```

7.5 De novo Motif Discovery III – FlexModule

Both *flexmodule_motif* and *flexmodule_tum* are special forms of a more general sampler *FlexModule*. *FlexModule* allows users to specify module structures, and the motif search will be biased towards user-specified module structures. The model and usage of *FlexModule* will be discussed elsewhere.

8. KNOWN MOTIF MAPPING TOOLBOX – MAPPING TRANSCRIPTION FACTOR BINDING SITES FOR KNOWN MOTIFS

8.1 Introduction

When the binding motif of a transcription factor is known, the motif can be used to scan genomic sequences to predict transcription factor binding sites (TFBS). This section will introduce various functions for known motif mapping.

8.2 List of Functions

(1) *motifmap_consensusscan_genome*, *motifmap_consensusscan* – Mapping a consensus motif to genomic regions or FASTA sequences.

(2) *motifmap_matrixscan_genome*, *motifmap_matrixscan* – Mapping a position specific weight matrix (PWM) to genomic regions or FASTA sequences.

(3) *motifmap_filter_genome* – Filtering TFBS by conservation and protein coding characteristics.

(4) *motifmap_getsitearound* – Extending TFBS to include flanking regions.

(5) *motifmap_getsitearoundcs* – Getting average conservation scores for positions within and around TFBS.

(6) *motifmap_matrixscan_summary* – Computing relative enrichment level for a list of PWMs in target regions as compared to control regions.

(7) *motifmap_matrixscan_enrich* – Computing relative enrichment level of a PWM in ranked and tiered regions.

8.3 TFBS mapping I – Mapping a Consensus Motif to Genomic Regions or FASTA sequences

```
[Usage]
motifmap_consensusscan_genome -m [File that specifies the consensus motif] -gd [Path where genome sequences (*.sq files) are stored] -i [File that specifies genomic coordinates of target regions] -o [Output file] -mc [Maximum consensus mismatches allowed] -md [Maximum degenerate mismatches allowed] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored]

motifmap_consensusscan -m [File that specifies the consensus motif] -d [Directory where input sequences and conservation scores are stored] -i [FASTA file that contains input sequences] -o [Output File] -mc [Maximum consensus mismatches allowed] -md [Maximum degenerate mismatches allowed] -c [Conservation cutoff] -ch [Prefix of the files that contain conservation scores]
```

motifmap_consensusscan_genome or *motifmap_consensusscan* can be used to map a consensus motif to specified genomic regions or FASTA sequences. One can use *motifmap_consensusscan_genome* to map a motif to specified genomic regions, and use *motifmap_consensusscan* to map a motif to FASTA sequences. The consensus motif should be given in MOTIF_CONS format (Appendix A.1.9), e.g., "TGGGT[A]GGTC[G,T]". In the above example, "TGGGTGGTC" is defined as the *consensus sequence*, and "TGGGT[A]GGTC[G,T]" is defined as the *degenerate consensus sequence*. A binding site "TGGGAGGTA" has 2 mismatches to the consensus (or 2 *consensus mismatches*), and 1 mismatch to the degenerate consensus (or 1 *degenerate mismatch*). The 2 consensus mismatches are TGGG**A**GGT**A**, and the 1 degenerate mismatch is TGGGAGGT**A**.

After specifying the maximum consensus mismatches and degenerate mismatches allowed, the motif will be used to scan the genomic sequences. A site will be reported as a TFBS if both of the following conditions are satisfied:

(i) # of consensus mismatches <= [Maximum consensus mismatches allowed];

(ii) # of degenerate mismatches <= [Maximum degenerate mismatches allowed].

One can choose to filter TFBS further by cross-species conservation. In order to do so, one needs to specify "-c" and "-cd" (or "-ch") options to define a conservation cutoff and conservation scores to use. A TFBS, if its average conservation score < [Conservation cutoff], will be filtered out. Only those TFBS whose average conservation scores >= [Conservation cutoff] will be reported.

Each reported TFBS will be assigned a site score. The site score is defined as:

```
Site score = - [# of degenerate mismatches] - 0.1*[# of consensus mismatches]
```

8.3.1 *motifmap_consensusscan_genome*

motifmap_consensusscan_genome can be used as follows.

(1) Make sure that relevant local genome databases are available.

(2) Prepare a file in MOTIF_CONS format (Appendix A.1.9) that specifies the consensus motif, e.g., Gli_cons.txt.

(3) Prepare a file in COD_C format (Appendix A.1.6) that specifies genomic regions to which the motif will be mapped, e.g., coordinates.txt.

(4) Run *motifmap_consensusscan_genome*. For example:

```
> motifmap_consensusscan_genome -m Gli_cons.txt -gd /data/genomes/human/hg17/ -i coordinates.txt -o Gli.map -mc 2 -md 0 -c 40 -cd /data/genomes/human/hg17/conservation/phastcons/
```

In *motifmap_consensusscan_genome,*

"-m" specifies the file that contains the consensus motif in MOTIF_CONS format.

"-gd" specifies the directory where genome sequences (*.sq files) are stored. Make sure that the directory contains two other files: chrlist.txt and chrlen.txt.

"-i" specifies the file that contains target genomic regions in COD_C format. The motif will be mapped to these regions.

"-o" specifies the output file that will save mapping results.

"-mc" specifies the maximum consensus mismatches allowed.

"-md" specifies the maximum degenerate mismatches allowed.

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

"-c" specified the conservation cutoff. If one does not want to apply conservation filter, please do not include "-c" and "-cd" options in the command.

(5) After running *motifmap_consensusscan_genome*, the mapping results will be saved to two files. In the above example, the two files are "Gli.map" and "Gli.map.stat" respectively. "Gli.map" is a file in MOTIF_SITE format (Appendix A.1.11) that reports all TFBS obtained from the scan. "Gli.map.stat" is a text file that contains several summary statistics. Below is an example (*.map.stat):

```
EffecLen= 1444533834
TotalSite= 735130
ConsLen= 345573331
```

Here, "EffecLen" is the total number of non-repeat base pairs in the scanned regions. "TotalSite" is the total number of reported TFBS. "ConsLen" is the total number of non-repeat positions that can pass the conservation cutoff in the scanned regions.

8.3.2 *motifmap_consensusscan*

motifmap_consensusscan can be used as follows.

(1) Prepare a file in MOTIF_CONS format (Appendix A.1.9) that specifies the consensus motif, e.g., Gli_cons.txt.

(2) Prepare a FASTA file (Appendix A.1.4) that contains DNA sequences to be scanned, e.g., Gli_reg.fa.

(3) If one needs to filter TFBS by cross-species conservation, for each sequence in the FASTA file, prepare a separate file in CS format (Appendix A.1.2) that stores the sequence's conservation scores. These CS files should be stored in the same directory as the FASTA file. The size of each CS file should be equal to the length of its corresponding sequence. The name of a CS file should have the following format: [Prefix][Index]_[Sequence Name].cs. Here, [Prefix] is a title that is the same for all CS files associated with a FASTA file. [Index] is an integer used to index sequences in the FASTA file. The index is 0-based, i.e., the first sequence is indexed by 0, the second sequence is indexed by 1, etc. [Sequence Name] should be the same as the name following ">" characters in the FASTA file. For example, if the input FASTA file is Gli_reg.fa:

```
>seq1
TTTCAATGTGTCCTAACTGTTTGGAATAAATCTAAGGTTGTCCCTAGTTGTCATGGCATT
>seq2
CAACCCTTTTCAATGGACTATCTCTTCATTCATTTCTGAATCCGTCCACAATATTAAGGA
```

then one should prepare two CS files, named Gli_reg_0_seq1.cs and Gli_reg_1_seq2.cs respectively.

(4) Run *motifmap_consensusscan*. For example:

```
> motifmap_matrixscan -m Gli_cons.txt -d /users/ -i Gli_reg.fa -o Gli.map
-mc 2 -md 0 -c 40 -ch Gli_reg_
```

In *motifmap_consensusscan*,

"-m" specifies the file that contains the consensus motif in MOTIF_CONS format.

"-d" specifies the directory where FASTA sequences and conservation scores (i.e., CS files) are stored.

"-i" specifies the name of the input FASTA file. Since the directory information is already given by "-d", there is no need to give the full path of the file. Please only specify the file name here.

"-o" specifies the output file that will save mapping results.

"-mc" specifies the maximum consensus mismatches allowed.

"-md" specifies the maximum degenerate mismatches allowed.

"-ch" specifies the [Prefix] attached to the CS files. It will be used to find and access conservation score files.

“-c” specified the conservation cutoff. If one does not want to apply conservation filter, please do not include “-c” and “-ch” options in the command.

(5) After running *motifmap_consensusscan*, the mapping results will be saved to two files. In the above example, the two files are “Gli.map” and “Gli.map.stat” respectively. “Gli.map” is a file in MOTIF_SITE format (Appendix A.1.11) that reports all TFBS obtained from the scan. “Gli.map.stat” is a text file that contains several summary statistics. Below is an example (*.map.stat):

```
EffecLen= 1444533834
TotalSite= 735130
ConsLen= 345573331
```

Here, “EffecLen” is the total number of non-repeat base pairs in the scanned regions. “TotalSite” is the total number of reported TFBS. “ConsLen” is the total number of non-repeat positions that can pass the conservation cutoff in the scanned regions.

8.4 TFBS mapping II – Mapping a Position Specific Weight Matrix (PWM) to Genomic Regions of FASTA sequences

[Usage]

```
motifmap_matrixscan_genome -m [File that specifies the PWM]  -gd [Path where genome sequences (*.sq files) are stored] -i [File that specifies genomic coordinates of target regions] -o [Output file] -r [Likelihood ratio cutoff] -b [Order of Markov background model] -bt [Method to fit Markov background: region or genome] -bd [Path where genome-wide Markov background models are stored] -bs [Step size for fitting genome-wide Markov background models] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored]
```

```
motifmap_matrixscan -m [File that specifies the PWM] -d [Directory where input sequences and conservation scores are stored] -i [FASTA file that contains input sequences] -o [Output File]-r [Likelihood ratio cutoff] -b [Order of Markov background model] -c [Conservation cutoff] -ch [Prefix of the files that contain conservation scores]
```

motifmap_matrixscan_genome or *motifmap_matrixscan* can be used to map a PWM to specified genomic regions or FASTA sequences. One can use *motifmap_matrixscan_genome* to map a motif to specified genomic regions, and use *motifmap_matrixscan* to map a motif to FASTA sequences. The motif PWM should be given in MOTIF_MAT format (Appendix A.1.10). The PWM will be used to scan the genomic sequences. At each position, the likelihood of the motif model (i.e., the probability to generate a site by the PWM) is compared to the likelihood of a background model (i.e., the probability to generate a site by a Markov background). A site will be reported as a TFBS if the likelihood ratio between the motif model and the background model >= [Likelihood ratio cutoff].

One can choose to filter TFBS further by cross-species conservation. In order to do so, one needs to specify “-c” and “-cd” (or “-ch”) options to define a conservation cutoff and to choose conservation scores to use. A TFBS, if its average conservation score < [Conservation cutoff], will be filtered out. Only those TFBS whose average conservation scores >= [Conservation cutoff] will be reported.

Each reported TFBS will be assigned a site score. The site score is defined as:

```
Site score = Log10(Likelihood ratio between the motif model and the background model).
```

Users can specify the order of Markov background models through the “-b” option. In *motifmap_matrixscan_genome*, one can also specify the way to fit the background models though the “-bt” option. If “-bt region”, then the Markov background will be fitted using the input genomic regions or FASTA sequences. If “-bt genome”, then the Markov models pre-computed in the process of establishing local genome databases will be used.

In fitting background Markov models, two types of models are fitted, a forward model and a backward model. For a sequence ABCD, a forward 3rd order Markov model fits transition probability P_f(ABC->D), whereas a backward 3rd order Markov model fits transition probability P_b(BCD->A).

When computing likelihood ratio between the motif model and the background model, both forward and backward background likelihood are computed. For a site CCT<u>GGGTG</u>GTC, the forward background likelihood of the underscored part is L_{Bf}=P_f(CCT->G)*P_f(CTG->G)*P_f(TGG->G)*P_f(GGG->T)*P_f(GGT->G), and the backward background likelihood is L_{Bb}=P_b(GTC->G)*P_b(GGT->T)*P_b(TGG->G)* P_b(GTG->G)*P_b(GGT->G). The motif likelihood is also computed for both forward strand (L_{M+}) and reverse complement strand (L_{M-}). The motif likelihood is then compared to both L_{Bf} and L_{Bb}. A site is called a TFBS in the “+” strand, if

min(L_{M+}/L_{Bf}, L_{M+}/L_{Bb})>=[Likelihood ratio cutoff].

A site is called a TFBS in the “-” strand, if

min(L_{M-}/L_{Bf}, L_{M-}/L_{Bb})>=[Likelihood ratio cutoff].

8.4.1 *motifmap_matrixscan_genome*

motifmap_matrixscan_genome can be used as follows.

(1) Make sure that relevant local genome databases are available.

(2) Prepare a file in MOTIF_MAT format (Appendix A.1.10) that specifies the motif PWM, e.g., Gli_mat.txt.

(3) Prepare a file in COD_C format (Appendix A.1.6) that specifies genomic regions to which the motif will be mapped, e.g., Gli_coord.txt.

(4) Run *motifmap_matrixscan_genome*. For example:

```
>motifmap_matrixscan_genome -m Gli_mat.txt -gd
/data/genomes/human/hg17/ -i Gli_coord.txt -o Gli.map -r 500 –b 3 –bt
genome –bd /data/genomes/human/hg17/markovbg/S100000_W1000000/ -bs
100000 -c 40 -cd /data/genomes/human/hg17/conservation/phastcons/
```

In *motifmap_matrixscan_genome*,

“-m” specifies the file that contains the motif PWM in MOTIF_MAT format.

“-gd” specifies the directory where genome sequences (*.sq files) are stored. Make sure that the directory contains two other files: chrlist.txt and chrlen.txt.

“-i” specifies the file that contains target genomic regions in COD_C format. The motif will be mapped to these regions.

“-o” specifies the output file that will save mapping results.

“-r” specifies the likelihood ratio cutoff.

“-b” specifies the order of background Markov model.

“-bt” specifies the method to fit the background Markov model.

If “-bt region”, then the background model will be fitted using the input genomic regions which are specified by “-i” option. In this case, “-bd” and “-bs” options shouldn’t be used anymore.

If”-bt genome”, then the program will use the pre-computed Markov background model stored in local genome databases. These models are computed using *motifmap_matrixscan_genome_bg* when establishing the local genome databases (refer to section 2.6). Different genomic loci may have different transition probability matrices. If “-bt genome” is used, one must also specify “-bd” and “-bs” options in the command.

“-bd” specifies the directory where pre-computed genome Markov background models are stored. For example, if such models are computed for human (hg17) by *motifmap_matrixscan_genome_bg* using a step size S=100000 and a window size W=1000000, they usually will be stored in a directory such as /data/genomes/human/hg17/markovbg/S100000_W1000000/. The 3rd order models are stored in /data/genomes/human/hg17/markovbg/S100000_W1000000/3/, and the 0th order models are stored in /data/genomes/human/hg17/markovbg/S100000_W1000000/0/. If one wants to use the 3rd order Markov model as the background, then “-bd” and “-bs” should be specified as “-b 3 –bt genome -bd /data/genomes/human/hg17/markovbg/S100000_W1000000/ -bs 100000”.

“-bs” specifies the step size S used to compute the genome Markov background models (refer to section 2.6).

“-cd” specifies the directory where genome conservation scores (*.cs files) are stored.

“-c” specified the conservation cutoff. If one does not want to apply conservation filter, please do not include “-c” and “-cd” options in the command.

(5) After running *motifmap_matrixscan_genome*, the mapping results will be saved to two files. In the above example, the two files are “Gli.map” and “Gli.map.stat” respectively. “Gli.map” is a file in MOTIF_SITE format (Appendix A.1.11) that reports all TFBS obtained from the scan. “Gli.map.stat” is a text file that contains several summary statistics. Below is an example (*.map.stat):

```
EffecLen= 1444533834
TotalSite= 735130
ConsLen= 345573331
```

Here, “EffecLen” is the total number of non-repeat base pairs in the scanned regions. “TotalSite” is the total number of reported TFBS. “ConsLen” is the total number of non-repeat positions that can pass the conservation cutoff in the scanned regions.

8.4.2 *motifmap_matrixscan*

motifmap_matrixscan can be used as follows.

(1) Prepare a file in MOTIF_MAT format (Appendix A.1.10) that specifies the motif PWM, e.g., Gli_mat.txt.

(2) Prepare a FASTA file (Appendix A.1.4) that contains DNA sequences to be

scanned, e.g., Gli_reg.fa.

(3) If one needs to filter TFBS by cross-species conservation, for each sequence in the FASTA file, prepare a separate file in CS format (Appendix A.1.2) that stores the sequence's conservation scores. These CS files should be stored in the same directory as the FASTA file. The size of each CS file should be equal to the length of its corresponding sequence. The name of a CS file should have the following format: [Prefix][Index]_[Sequence Name].cs. Here, [Prefix] is a title that is the same for all CS files associated with a FASTA file. [Index] is an integer used to index sequences in the FASTA file. The index is 0-based, i.e., the first sequence is indexed by 0, the second sequence is indexed by 1, etc. [Sequence Name] should be the same as the name specified in the FASTA file, i.e., the string following ">" characters in the FASTA file. For example, if the input FASTA file is Gli_reg.fa:

```
>seq1
TTTCAATGTGTCCTAACTGTTTGGAATAAATCTAAGGTTGTCCCTAGTTGTCATGGCATT
>seq2
CAACCCTTTTCAATGGACTATCTCTTCATTCATTTCTGAATCCGTCCACAATATTAAGGA
```

then one should prepare two CS files, named Gli_reg_0_seq1.cs and Gli_reg_1_seq2.cs respectively.

(4) Run *motifmap_matrixscan*. For example:

```
> motifmap_matrixscan -m Gli_mat.txt -d /users/ -i Gli_reg.fa -o Gli.map
-r 500 -b 3 -c 40 -ch Gli_reg_
```

In *motifmap_matrixscan*,

"-m" specifies the file that contains the motif PWM in MOTIF_MAT format.

"-d" specifies the directory where FASTA sequences and conservation scores (i.e., CS files) are stored.

"-i" specifies the name of the input FASTA file. Since the directory information is already given by "-d", there is no need to give the full path of the file. Please only specify the file name here.

"-o" specifies the output file that will save mapping results.

"-r" specifies the likelihood ratio cutoff.

"-b" specifies the order of background Markov model.

"-ch" specifies the [Prefix] attached to the CS files. It will be used to find and access conservation score files.

"-c" specified the conservation cutoff. If one does not want to apply conservation filter, please do not include "-c" and "-ch" options in the command.

(5) After running *motifmap_matrixscan*, the mapping results will be saved to two files in a similar way as *motifmap_matrixscan_genome*.

8.5 TFBS mapping III – Filtering TFBS by conservation and protein coding characteristics

```
[Usage]
motifmap_filter_genome -i [File that contains input TFBS] -o [File to save filtered TFBS] -c [Conservation cutoff] -cm [File that specifies conservation mask] -cd [Path where genome conservation scores (*.cs files) are stored] -cds [Non-CDS ratio] -cdsd [Path where genome protein coding indicators (*.cds files) are stored
```

motifmap_filter_genome can be used to filter TFBS by their cross-species conservation and protein coding characteristics. Users can choose to check conservation characteristics of certain positions in the TFBS, say the 1st, 2nd, 3rd, 6th, 7th and 8th position of a motif CCCATGGG. The chosen position is defined by a conservation mask defined as “1 1 1 0 0 1 1 1”. A TFBS will be filtered out if one of the following conditions is satisfied:

(i) One of the chosen positions in the conservation mask has a conservation score < [conservation cutoff]

(ii) The fraction of non-protein-coding base pairs in the TFBS < [Non-CDS ratio].

In order to use *motifmap_filter_genome*, please follow the steps below.

(1) Make sure that the relevant local genome databases are available.

(2) Prepare a file in MOTIF_SITE format that contains all TFBS to be processed. All TFBS should be sites of the same motif and should have a common length.

(3) Prepare a file that specifies the conservation mask in MOTIF_MASK format (Appendix A.1.12).

(4) Run *genome_filtergenome*. For example:

```
> motifmap_filter_genome -i Gli.map -o Gli_filtered.map -c 250 -cm Gli_mask.txt -cd /data/genomes/human/hg17/conservation/phastcons/ -cds 0.9 -cdsd /data/genomes/human/hg17/cds/
```

In *genome_filtergenome*,

“-i” specifies the file that contains input TFBS in MOTIF_SITE format.

“-o” specifies the output file that will save filtered TFBS in MOTIF_SITE format.

“-cd” specifies the directory where genome conservation scores (*.cs files) are stored.

“-cm” specifies the file that contains the conservation mask in MOTIF_MASK format.

“-c” specified the conservation cutoff. If one does not want to apply conservation filter, please do not include “-c”, “-cd” and “-cm” options in the command.

“-cdsd” specifies the directory where genome CDS indicators (*.cds files) are stored.

“-cds” specified the non-CDS cutoff. If one does not want to apply protein coding filter, please do not include “-cds” and “-cdsd” options in the command.

8.6 TFBS mapping IV – Extending TFBS to Cover Flanking Regions

[Usage]

```
motifmap_getsitearound -i [File that contains input TFBS] -w [Extension
Window Size] -d [Path where genome sequences (*.sq files) are stored] -s
[species name, e.g., human, mouse, dog] -cn [Output file format] -a [Output
index system] -o [Output File]
```

motifmap_getsitearound can be used to extend TFBS to cover flanking regions. This function takes a file in MOTIF_SITE format as input. After specifying a window size W, each TFBS in the input file will be extended W base pairs toward both ends. In other words, if the coordinates of a TFBS is "chrN:Start~End", then the extended region is "chrN:(Start-W)~(End+W)". The extended regions will be exported to a file in COD_C or COD_N format. These extended regions can be used in various downstream analyses such as searching for cofactors of a given motif.

In order to use *motifmap_getsitearound*, please follow the steps below.

(1) Make sure that the relevant local genome databases are available. Indeed, only two files in the database will be used here: chrlist.txt and chrlen.txt. Therefore make sure that these two files are available.

(2) Prepare a file in MOTIF_SITE format (Appendix A.1.11) that contains all input TFBS, e.g., Gli.map

(3) Run *motifmap_getsitearound*. For example:

```
> motifmap_getsitearound -i Gli.map -w 200 -d /data/genomes/human/hg17/
-s human -cn 1 -a 1 -o Gli_extended.txt
```

In *motifmap_getsitearound*,

"-i" specifies the input file that contains all TFBS to be processed. The file is in MOTIF_SITE format.

"-w" specifies the extension window size W.

"-d" specifies the directory where genome sequences (*.sq files) are stored.

"-s" specifies the species name, e.g., human, mouse, dog, cow, chicken, zebrafish.

"-cn" specifies the format of the output file. If "-cn 0", then the extended regions will be saved to a file in COD_C format. If "-cn 1", then the extended regions will be saved to a file in COD_N format.

"-a" specifies the index system of the output file. If "-a 0", then the first column in the input file will be kept in its original form in the output file. If "-a 1", then the output regions will be re-indexed. The first region will be indexed as 0, the second region will be indexed as 1, and so on.

"-o" specifies the output file that will save extended regions in COD_C or COD_N format.

8.7 TFBS mapping V – Computing Average Conservation Scores for TFBS and Flanking Positions

```
[Usage]
motifmap_getsitearoundcs -i [File that contains input TFBS] -o [Output
File] -l [Motif Length] -w [Flanking Window Size] -s [species name, e.g.,
human, mouse, dog] -gd [Path where genome sequences (*.sq files) are stored]
-cd [Path where genome conservation scores (*.cs files) are stored]
```

motifmap_getsitearoundcs can be used to get average conservation scores for each position within and flanking the binding sites of a given motif. For example, if a motif is mapped to human genome and 1000 TFBS are obtained from the mapping. For each individual position within and flanking the motif, the function will compute the mean conservation scores of the 1000 sites and export the average to the output file. As a result, in the output file, a mean conservation score will be attached to each position within and around the motif.

In order to use *motifmap_getsitearoundcs*, please follow the steps below.

(1) Make sure that the relevant local genome databases are available.

(2) Prepare a file in MOTIF_SITE format (Appendix A.1.11) that contains all input TFBS, e.g., Gli.map. Make sure that all TFBS have the same length.

(3) Run *motifmap_getsitearoundcs*. For example:

```
> motifmap_getsitearoundcs -i Gli.map -o Gli_cscurve.txt -l 12 -w 50 -s
human -gd /data/genomes/human/hg17/ -cd
/data/genomes/human/hg17/conservation/phastcons/
```

In *motifmap_getsitearoundcs*,

"-i" specifies the input file that contains all TFBS to be processed. The file is in MOTIF_SITE format.

"-o" specifies the output file that will save average conservation scores.

"-l" specifies the motif length (i.e., the length of TFBS).

"-w" specifies the flanking window size W. Average conservation scores will be computed for each position within the interval [Motif_START-W, Motif_END+W].

"-s" specifies the species name, e.g., human, mouse, dog, cow, chicken, zebrafish.

"-gd" specifies the directory where genome sequences (*.sq files) are stored.

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

(4) After running *motifmap_getsitearoundcs*, the output file should be a text file that contain l+2*w lines. Each line corresponds to a position in the interval [Motif_START-W, Motif_END+W]. There is a single number in each line which is the average conservation score for the corresponding position.

8.8 TFBS mapping VI – Computing Relative Enrichment Level for a List of PWMs

```
[Usage]
motifmap_matrixscan_genome_summary -mr [File that contains the list of motifs and likelihood ratios] -gd [Path where genome sequences (*.sq files) are stored] -i [File that specifies genomic coordinates of target regions] -n [File that specifies genomic coordinates of control regions] -o [Output file] -b [Order of Markov background model] -bt [Method to fit Markov background: region or genome] -bd [Path where genome-wide Markov background models are stored] -bs [Step size for fitting genome-wide Markov background models] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored]
```

motifmap_matrixscan_genome_summary can be used to compare the enrichment levels of a list of motifs (PWM) in target genomic regions with their enrichment levels in control genomic regions. For each motif, the relative enrichment level will be computed.

The relative enrichment level is characterized by three statistics, r_1 , r_2 and r_3. Assume that n_{1B} counts how many times a motif occur in target regions, n_{2B} is the total length of non-repeat sequences in target regions, n_{1C} counts how many times the motif occur in control regions, and n_{2C} is the total length of non-repeat sequences in control regions. $r_1=(n_{1B}/n_{2B})/(n_{1C}/n_{2C})$ defines the relative enrichment level of the motif. Similarly, let n_{3k} (k=B or C) count the number of phylogenetically conserved motif sites in specified genomic regions, and n_{4k} count phylogenetically conserved non-repeat base pairs in the regions. $r_2=(n_{3B}/n_{4B})/(n_{3C}/n_{4C})$ then defines motif's relative enrichment level in phylogenetically conserved regions. Notice that n_{3k}/n_{1k} is the percentage of motif sites that are conserved, and n_{4k}/n_{2k} is the percentage of genomic sequences that are conserved. Finally, r_3, defined as $(n_{3B}/n_{2B})/(n_{3C}/n_{2C})$, characterizes the relative enrichment level of phylogenetically conserved sites in target regions (not necessary conserved). Notice that $r_3/r_2=(n_{4B}/n_{2B})/(n_{4C}/n_{2C})$ characterizes whether or not the target regions tend to be more phylogenetically conserved than control regions.

In order to use *motifmap_matrixscan_genome_summary*, please follow the steps below.

(1) Make sure that the relevant local genome databases are available.

(2) Prepare a file in MOTIF_LIST format (Appendix A.1.13) that contains all motifs to be processed and their corresponding likelihood ratio cutoffs, e.g., motiflist.txt. Make sure that the motifs included in the file exist in appropriate directories.

(3) Prepare a file in COD_C (Appendix A.1.6) format that specifies target genomic regions, e.g., target_coord.txt.

(4) Prepare a file in COD_C format that specifies control genomic regions, e.g., control_coord.txt.

(5) Run *motifmap_matrixscan_genome_summary*. For example:

```
> motifmap_matrixscan_genome_summary -mr motiflist.txt -gd /data/genomes/human/hg17/ -i target_coord.txt -n control_coord.txt -o motifs_enrich.txt -b 3 -bt genome -bd /data/genomes/human/hg17/markovbg/S100000_W1000000 -bs 100000 -c 40 -cd /data/genomes/human/hg17/conservation/phastcons/
```

In *motifmap_matrixscan_genome_summary,*

"-mr" specifies the file that contains the list of motifs to be processed and their likelihood ratio cutoffs. The file is in MOTIF_LIST format.

"-gd" specifies the directory where genome sequences (*.sq files) are stored.

"-i" specifies the file that contains target genomic regions in COD_C format.

"-n" specifies the file that contains control genomic regions in COD_C format.

"-o" specifies the output file that will save motifs' relative enrichment levels in MOTIF_SUMMARY (Appendix A.1.14) format.

. "-b" specifies the order of background Markov model used in motif mapping.

"-bt" specifies the method to fit the background Markov model.

If "-bt region", then the background model will be fitted using the target and control genomic regions. In this case, "-bd" and "-bs" options shouldn't be used anymore.

If"-bt genome", then the program will use the pre-computed Markov background model stored in local genome databases. These models are computed using *motifmap_matrixscan_genome_bg* when establishing the local genome databases (refer to section 2.6). Different genomic loci may have different transition probability matrices. If "-bt genome" is used, one must also specify "-bd" and "-bs" options in the command.

"-bd" specifies the directory where pre-computed genome Markov background models are stored. For example, if such models are computed for human (hg17) by *motifmap_matrixscan_genome_bg* using a step size S=100000 and a window size W=1000000, they usually will be stored in a directory such as /data/genomes/human/hg17/markovbg/S100000_W1000000/. The 3rd order models are stored in /data/genomes/human/hg17/markovbg/S100000_W1000000/3/, and the 0th order models are stored in /data/genomes/human/hg17/markovbg/S100000_W1000000/0/. If one wants to use the 3rd order Markov model as the background, then "-bd" and "-bs" should be specified as "-b 3 –bt genome -bd /data/genomes/human/hg17/markovbg/S100000_W1000000/ -bs 100000".

"-bs" specifies the step size S used to compute the genome Markov background models (refer to section 2.6).

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

"-c" specified the conservation cutoff. If one does not want to apply conservation filter, please do not include "-c" and "-cd" options in the command.

8.9 TFBS mapping VII – Computing Relative Enrichment Level of a PWM in Tiered Regions

`[Usage]`
`motifmap_matrixscan_genome_enrich -m [File that specifies the PWM] -gd [Path where genome sequences (*.sq files) are stored] -i [File that specifies genomic coordinates of target regions] -n [File that specifies genomic coordinates of control regions] -s [tier size] -o [Output file] -r [Likelihood ratio cutoff] -b [Order of Markov background model] -bt [Method to fit Markov background:`** ***`region`*** **`or`** ***`genome`*** **`] -bd [Path where genome-wide Markov background models are stored] -bs [Step size for fitting genome-wide Markov background models] -c [Conservation cutoff] -cd [Path where genome conservation scores (*.cs files) are stored]`

motifmap_matrixscan_genome_enrich can be used to compute the relative enrichment levels of a PWM in target regions as compared to control regions. The target regions should be ranked before applying this function. The ranked target regions will be grouped into several tiers based on their ranking, and relative enrichment level will be computed for each tier. This function can be used to examine the change of enrichment levels when the region quality is decreasing. The relative enrichment level is defined in section 8.8.

In order to use *motifmap_matrixscan_genome_enrich*, please follow the steps below.

(1) Make sure that the relevant local genome databases are available.

(2) Prepare a file in MOTIF_MAT format (Appendix A.1.10) that specifies the motif PWM, e.g., Gli_mat.txt.

(3) Prepare a file in COD_C (Appendix A.1.6) format that specifies target genomic regions, e.g., target_coord.txt.

(4) Prepare a file in COD_C format that specifies control genomic regions, e.g., control_coord.txt.

(5) Run *motifmap_matrixscan_genome_enrich*. For example:

```
> motifmap_matrixscan_genome_enrich -m Gli_mat.txt -gd /data/genomes/human/hg17/ -i target_coord.txt -n control_coord.txt -s 20 -o Gli_tierenrich.txt -r 500 -b 3 -bt genome -bd /data/genomes/human/hg17/markovbg/S100000_W1000000 -bs 100000 -c 40 -cd /data/genomes/human/hg17/conservation/phastcons/
```

In *motifmap_matrixscan_genome_enrich*,

"-m" specifies the file that contains the motif PWM in MOTIF_MAT format.

"-gd" specifies the directory where genome sequences (*.sq files) are stored.

"-i" specifies the file that contains target genomic regions in COD_C format.

"-n" specifies the file that contains control genomic regions in COD_C format.

"-s" specifies the tier size, i.e., how many regions are contained in each tier.

"-o" specifies the output file that will save motifs' relative enrichment levels in

MOTIF_TIERENRICH (Appendix A.1.15) format.

"-r" specifies the likelihood ratio cutoff for declaring TFBS (refer to section 8.4 for relevant discussions).

. "-b" specifies the order of background Markov model used in motif mapping.

"-bt" specifies the method to fit the background Markov model.

If "-bt region", then the background model will be fitted using the target and control genomic regions. In this case, "-bd" and "-bs" options shouldn't be used anymore.

If"-bt genome", then the program will use the pre-computed Markov background model stored in local genome databases. These models are computed using *motifmap_matrixscan_genome_bg* when establishing the local genome databases (refer to section 2.6). Different genomic loci may have different transition probability matrices. If "-bt genome" is used, one must also specify "-bd" and "-bs" options in the command.

"-bd" specifies the directory where pre-computed genome Markov background models are stored. For example, if such models are computed for human (hg17) by *motifmap_matrixscan_genome_bg* using a step size S=100000 and a window size W=1000000, they usually will be stored in a directory such as /data/genomes/human/hg17/markovbg/S100000_W1000000/. The 3rd order models are stored in /data/genomes/human/hg17/markovbg/S100000_W1000000/3/, and the 0th order models are stored in /data/genomes/human/hg17/markovbg/S100000_W1000000/0/. If one wants to use the 3rd order Markov model as the background, then "-bd" and "-bs" should be specified as "-b 3 –bt genome -bd /data/genomes/human/hg17/markovbg/S100000_W1000000/ -bs 100000".

"-bs" specifies the step size S used to compute the genome Markov background models (refer to section 2.6).

"-cd" specifies the directory where genome conservation scores (*.cs files) are stored.

"-c" specified the conservation cutoff. If one does not want to apply conservation filter, please do not include "-c" and "-cd" options in the command.

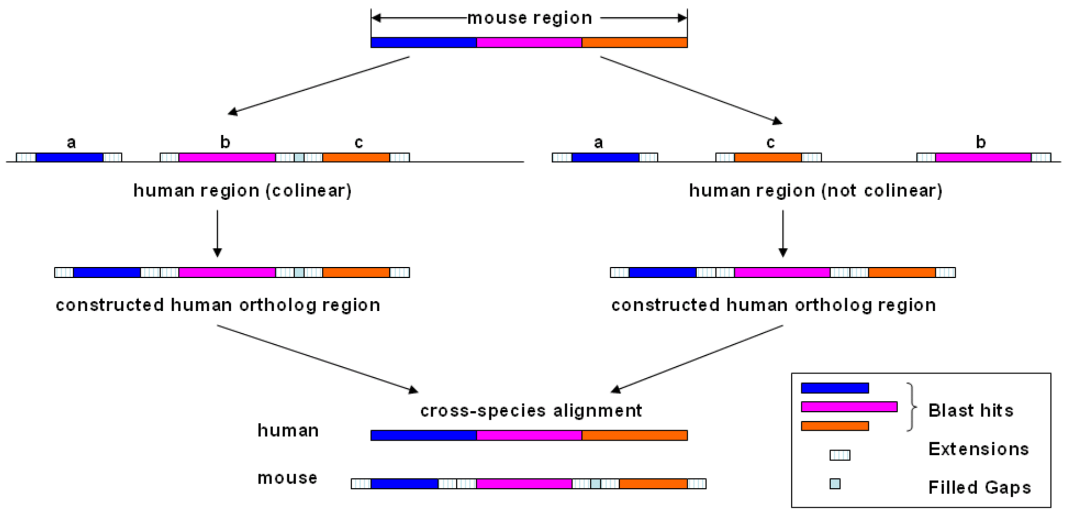
9. CROSS-SPECIES COMPARISON TOOLBOX – CROSS-SPECIES ALIGNMENT AND TFBS ANNOTATION

9.1 Introduction

This section will introduce how to generate cross-species alignments of specific genomic regions, and how to map TFBS to the alignments for visualization. The resulting alignments can be used for detailed examination of the module structures of *cis*-regulatory regions, and can be used to help selecting regions for transgenic studies.

We first use an example to illustrate the basic procedure for constructing the cross-species alignment and mapping TFBS to the alignment, as well as the final result one will get. We then discuss how these jobs can be done in following sections.

In the example, a list of mouse genomic regions is given. We want to generate cross-species alignment for each of the region (see the figure below) and annotate the alignments by TFBS mapping. In order to do so, we will go through the following steps.



(1) Each mouse region will be associated to a gene. The gene that is closest to the region in the genome will be picked up.

(2) Orthologs of the mouse gene in other species will be identified. Each ortholog gene, say, the human ortholog, will be extended certain base pairs towards both ends. We will then search for human segments that are orthologous to the original mouse region in the extended human window.

(3) To find the orthologous human segments, the original mouse region will be aligned to the extended human window by BLAST. Any BLAST hit that has (i) E-value <= [E-value cutoff], and (ii) alignment length >= [Minimal Length], and (iii) alignment percent identity >= [Minimal Identity] will be kept for further processing.

(4) All BLAST hits will be ranked by their alignment quality, defined as the number of identical base pairs in the alignment. If there is a co-linearity requirement by users (i.e., “[Colinearity] = 1” in the parameter file), then BLAST hits will be checked one by one from the highest quality hit to the lowest quality hit. Any hit that

is not collinear with a hit with higher quality will be discarded. By “collinear”, we mean that the direction and relative position of two hits need to be consistent in the two species.

(5) Each BLAST hit will be extended certain base pairs to cover some flanking regions. The length of the extension on each side is equal to min{[Alignment Length]*[Extension Percentage], [Maximal Extension]}. For example, if [Maximal Extension]=100bp, [Extension Percentage]=0.25, and the alignment length of a BLAST hit is 100bp, then the hit will be extended min{100*0.25, 100}=25 base pairs on each side.

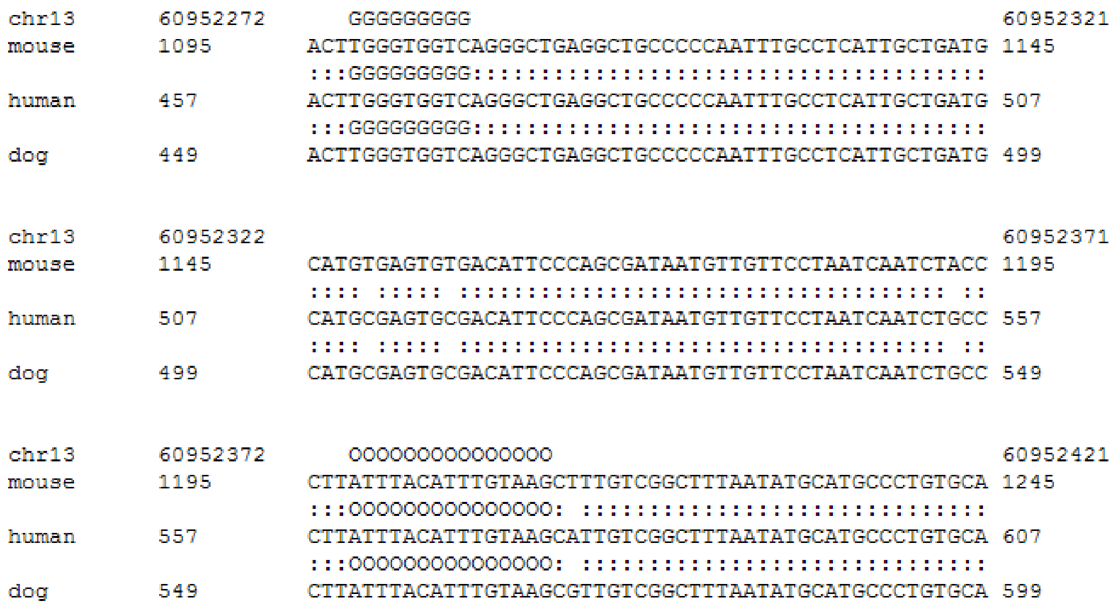
(6) After extension, if the distance between two BLAST hits <= [Maximal Link Gap], then the two hits will be merged into a single one. The merged region will include the gap (which will be called a “filled gap”) between the two hits.

(7) All the extended and merged BLAST hits will be linked together to form a sequence. This sequence will be treated as the human ortholog segment to the mouse region.

(8) Step 2~7 will be repeated for all other species.

(9) Ortholog regions from all species will be aligned by MLAGAN to generate the cross-species alignment.

(10) Transcription factor binding motifs specified by users will be mapped to the cross-species alignment. The final results will look like:



```
chr13      60952272      GGGGGGGGG                                        60952321
mouse      1095       ACTTGGGTGGTCAGGGCTGAGGCTGCCCCCAATTTGCCTCATTGCTGATG 1145
                      :::GGGGGGGGG::::::::::::::::::::::::::::::::::::::
human      457        ACTTGGGTGGTCAGGGCTGAGGCTGCCCCCAATTTGCCTCATTGCTGATG 507
                      :::GGGGGGGGG::::::::::::::::::::::::::::::::::::::
dog        449        ACTTGGGTGGTCAGGGCTGAGGCTGCCCCCAATTTGCCTCATTGCTGATG 499


chr13      60952322                                                       60952371
mouse      1145       CATGTGAGTGTGACATTCCCAGCGATAATGTTGTTCCTAATCAATCTACC 1195
                      :::: ::::: :::::::::::::::::::::::::::::::::::: ::
human      507        CATGCGAGTGCGACATTCCCAGCGATAATGTTGTTCCTAATCAATCTGCC 557
                      :::: ::::: :::::::::::::::::::::::::::::::::::: ::
dog        499        CATGCGAGTGCGACATTCCCAGCGATAATGTTGTTCCTAATCAATCTGCC 549


chr13      60952372      OOOOOOOOOOOOOOO                                  60952421
mouse      1195       CTTATTTACATTTGTAAGCTTTGTCGGCTTTAATATGCATGCCCTGTGCA 1245
                      :::OOOOOOOOOOOOOOO: ::::::::::::::::::::::::::::::
human      557        CTTATTTACATTTGTAAGCATTGTCGGCTTTAATATGCATGCCCTGTGCA 607
                      :::OOOOOOOOOOOOOOO: ::::::::::::::::::::::::::::::
dog        549        CTTATTTACATTTGTAAGCGTTGTCGGCTTTAATATGCATGCCCTGTGCA 599
```

Here, letters such as ‘G’, ‘O’, etc. are put on top of each sequence to indicate the locations of TFBS. Each letter indicates a type of motif, different letters correspond to different motifs, e.g., ‘G’=Gli, ‘O’=Oct4, etc. The meaning of each letter is defined by users. The top sequence in the alignment corresponds to the reference species. Each remaining species will be compared to the reference species. At a given column, if a species has the identical base pair to the reference species, a dot “.” will be put on top of the sequence. If all species in the column share an identical base pair, then a colon ‘:’ will be put on top of each sequence.

9.2 List of Functions

(1) *malign_genome_prepareortholog* – Prepare the extended window surrounding ortholog genes.

(2) *malign_genome_blasthit* – Generate cross-species alignments.

(3) *malign_motifmap* – Mapping transcription factor binding motifs to alignments.

(4) *malign_modulemap* – Selecting modules that contain specific TFBS.

9.3 Cross-species Comparison – Prerequisites

In order to generate the cross-species alignments, one needs to download and install BLAST and MLAGAN first. CisGenome itself does not provide the alignment program. What we will do here is to use CisGenome to collect ortholog regions, and to call BLAST and MLAGAN to generate the alignments, and then use CisGenome to map TFBS to the alignments.

BLAST can be downloaded from http://ncbi.nlm.nih.gov/BLAST/download.shtml.

MLAGAN can be downloaded from http://lagan.stanford.edu/lagan_web/index.shtml.

Please follow their instructions to install them.

9.4 Cross-species Comparison Step I – Preparing Ortholog Regions

[Usage]
malign_genome_prepareortholog -i [Input file] -o [Output file] -n [species number] -sf [Skip the first line] -r [type of extension] -up [length of upstream extension] -down [length of downstream extension]

The first step to generate cross-species alignment for specified genomic regions is to collect ortholog genes in related species, and get a extended window surrounding each ortholog gene. This can be done as follows.

(1) Prepare a file in COD_C format (Appendix A.1.6), say, `target_coord.txt`, that contains the genomic regions one wants to study, e.g., ChIP-chip binding regions in mouse genome.

(2) Run *refgene_getnearestgene* (section 4.4) to associate each region with its closest gene. For example:

```
> refgene_getnearestgene -d /data/genomes/mouse/mm6/annotation/refFlat_sorted.txt –dt 1 –s mouse -i target_coord.txt -o target_gene.txt –r 0 –up 50000 –down 50000
```

(3) Run *refflex_getmultiortholog* (section 4.7) to get ortholog genes. For example:

```
> refflex_getmultiortholog -i target_gene.txt -c 6 -d orthologsetting.txt -o target_ortho
```

When running this function, the orthologsetting.txt should be set to contain exactly the same number of species one want to include in the alignment.

(4) After the ortholog genes are obtained, one needs to extend them by certain amount of base pairs. This can be done using *malign_genome_prepareortholog*. For example:

```
>malign_genome_prepareortholog -i target_ortho.nsomap -o target_foraln.txt -n 4 -sf 1 -r 0 -up 50000 -down 50000
```

In *malign_genome_prepareortholog,*

"-i" specifies the *.nsomap file generated by *refflex_getmultiortholog*. This is a file in REFGENE_ORTHOLOG format.

"-o" specifies the output file.

"-n" specifies species number. The *.nsomap file consists of repeating sections starting with ">". The number of non-empty lines following each ">" is the species number one should use here.

"-sf" specifies whether or not to ignore the second line following each ">". If "-sf 1", the second line will be ignored. If "-sf 0" the second line will be kept. The reason to ignore the second line is as follows. When identifying ortholog genes, say, for a mouse gene, sometimes one may want to search in not only human, dog, etc. but also mouse itself. Here, the mouse is included as a positive control to see if the program runs appropriately. As a result, mouse ortholog to mouse gene (which basically are the same thing) will be reported in the *.nsomap file. Of course this is something redundant, i.e., the same thing is reported twice. Thus one may want to skip one line to remove this redundancy. "-sf" option here is provided to facilitate related operations.

"-r", "-up", "-down" specifies how to get the extended window surrounding a ortholog gene. Assume that TSS = transcription start, TES = transcription end, CDSS = protein coding region start, CDSE = protein coding region end, "-up" = M_u, "-down" = M_d. Below are the meanings of different "-r" options.

(i) "-r 0": the extended window is from M_u base pairs upstream of TSS to M_d base pairs downstream from TES.

(ii) "–r 1": the interval is from M_u base pairs upstream of TSS to M_d base pairs downstream from TSS.

(iii) "–r 2": the interval is from M_u base pairs upstream of TES to M_d base pairs downstream from TES.

(iv) "–r 3": the interval is from M_u base pairs upstream of CDSS to M_d base pairs downstream from CDSE.

(v) "–r 4": the interval is from M_u base pairs upstream of CDSS to M_d base pairs downstream from CDSS.

(vi) "–r 5": the interval is from M_u base pairs upstream of CDSE to M_d base pairs downstream from CDSE.

(5) After running *malign_genome_prepareortholog*, one should get a file like the one below that specifies the search region.

```
>0
mouse   chr2    146644426       146645770       +       Nkx2-2  NM_010919
   2       -       146640088       146643295       146640931   146642954
   2      146640088,146642696,    146641493,146643295,
human   chr20   21389653       21492663       +
dog     chr24   4889196 4991997 +

>1
mouse   chr12   53347110        53348554        +       Nkx2-9  NM_008701
   12      -       53337817        53339084        53337817    53338869
   2      53337817,53338722,      53338089,53339084,
human   chr14   36068967       36171536       +
dog     chr8    17905441       18069531       +
```

In this file, the first line following “>” is the original genomic region one wants to study. Each line thereafter specifies an extended window in the ortholog gene. Each line contains a genomic region in COD_C format.

9.5 Cross-species comparison Step II – Creating Multiple Alignments

`[Usage]`
`malign_genome_blasthit [Parameter file]`

The second step to generate cross-species alignment for specified genomic regions is run *malign_genome_blasthit*. Given the file generated by *malign_genome_prepareortholog* (e.g., `target_foraln.txt`), this function will automatically identify segments that are orthologous to the original genomic regions and align the ortholog regions by calling MLAGAN.

Using the example file from the previous section:

```
>0
mouse   chr2    146644426       146645770       +       Nkx2-2  NM_010919
   2       -       146640088       146643295       146640931   146642954
   2      146640088,146642696,    146641493,146643295,
human   chr20   21389653       21492663       +
dog    chr24   4889196 4991997 +
```

malign_genome_blasthit will first search for segments that are orthologous to mouse chr2:146644426-146645770. These segments will be found by BLAST mouse chr2:146644426-146645770 to human chr20:21389653-21492663 and dog chr24:4889196-4991997. The orthologous regions identified will then be used to generate cross-species alignment.

In order to use *malign_genome_blasthit*, one needs to prepare a main parameter file (say, *malign_genome_arg_u.txt*) and a genome sequence setting file (say, *genomesetting3_mouse.txt*). Then one can run *malign_genome_blasthit* as follows:

```
> malign_genome_blasthit malign_genome_arg_u.txt
```

Next, we introduce the parameter files.

9.5.1 Main Parameter File

Below is a sample main parameter file:

```
[Path of Blast] = ~/Tools/BioinfoTools/blast/blast-2.2.12/bin
[E-value Cutoff] = 1e-5
[Mask Small Case] (0:No, 1:Yes) = 1
[Minimal Length] = 25
[Minimal Identity] = 0.6
[Colinearity] = 0
[Extension Percentage] = 0.25
[Maximal Extension] = 100
[Maximal Link Gap] = 100
```

```
[Keep Best vs. All] (0:Best, 1:All) = 1

[Path of MLAGAN] = ~/Tools/BioinfoTools/lagan/lagan12
[MLAGAN Parameter] = -tree "((mouse human) dog)"

[Working Directory] = .
[Export File] = target_3way
[Input File] = target_foraln.txt
[Genome Setting] = genomesetting_mouse.txt
[Number of Databases] = 2
[Target & Databases]
mouse
human
dog
```

In the file,

"[Path of Blast] =" specifies the directory where the executable BLAST is installed.

"[E-value Cutoff] =" specifies the maximum E-value allowed for BLAST hits.

"[Mask Small Case] (0:No, 1:Yes) =" specifies if the small letters a, c, g and t are treated as repeats or not. If users set 1 here, then the small letters will be masked as repeats. If users set 0 here, then small letters will be used in searching the alignment.

"[Minimal Length] =" specifies the minimal length of BLAST hits that will be processed.

"[Minimal Identity] =" specifies the minimal percent identity level of a BLAST hit in order to be processed further.

"[Colinearity] =" specifies whether BLAST hits need to be collinear. If users set 1 here, BLAST hits that do not satisfy the colinearity criterion will be discarded. If users set 0 here, no colinearity requirement will be set.

"[Extension Percentage] =" specifies the length of the extension for each hit. Each BLAST hit will be extended (alignment length)* [Extension Percentage] base pairs to both ends.

"[Maximal Extension] =" specifies the maximal extensions. IF (alignment length)* [Extension Percentage]>[Maximal Extension], only [Maximal Extension] base pairs will be extended on each ends.

"[Maximal Link Gap] =" specifies the maximum gap distance allowed to merge to BLAST hits. If the distance between two BLAST hits <= [Maximal Link Gap], the two hits will be merged into a single one.

"[Keep Best vs. All] (0:Best, 1:All) =": If users set 1 here, only the best BLAST hit will be kept for constructing cross-species alignment. If users set 0 here, all BLAST hits will be linked to form ortholog sequences, and ortholog sequences will be aligned to get cross-species alignment.

"[Path of MLAGAN] =" pecifies the directory where the executable MLAGAN is installed.

“[MLAGAN Parameter] =” specifies the phylogenetic tree structure used by MLAGAN.

“[Working Directory] =” specifies the directory where all input files and output files are stored.

“[Export File] =” specifies the title of the output file.

“[Input File] =” specifies the input file. This is usually the output file generated by *malign_genome_prepareortholog* (e.g., target_foraln.txt)

“[Genome Setting] =” specifies the genome sequence setting file. This file is in GENOME_SEQUENCE_SETTING format (Appendix A.1.22).

“[Number of Databases] =” specifies the number of species excluding the original species.

“[Target & Databases]” specifies the original species and all other species one want to align. The line should be followed by 1+[Number of Databases] lines, each containing a species name (e.g., mouse, human, dog). The same species name should be used in specifying [MLAGAN Parameter] and the genome sequence setting file.

9.5.2 Genome Sequence Setting File

This is a file in GENOME_SEQUENCE_SETTING format (Appendix A.1.22). See Appendix A.1.22 for an example. This file specifies where to find genome sequences for each individual species. Please make sure that the file contains the same species as will be used in the alignment, and the order of their appearance should be consistent with their orders set in “[Target & Databases]” section of the main parameter file.

9.6 Cross-species comparison Step III – Annotating Multiple Alignments using TFBS

```
[Usage]
malign_motifmap [Parameter file]
```

After generating cross-species alignment, the third step is to map transcription factor binding motifs to the alignment. This can be done using *malign_motifmap*. In order to run *malign_motifmap,*

(1) For each motif one wants to map, prepare a file that specifies the motif pattern, either in MOTIF_CONS (Appendix A.1.9) or MOTIF_MAT format (Appendix A.1.10).

(2) Prepare a parameter file (say, malign_motifmap_arg.txt) as will be described below.

(3) Run *malign_motifmap* as follows:

```
> malign_motifmap malign_motifmap_arg.txt
```

A sample parameter file is shown as below:

```
[Path of CisGenome] = ~/Projects/CisGenome_Project_unix/bin/
[Working Directory] = .
[File Header] = target_3way
[Number of Clusters] = 30
[Number of Species] = 3
[Species]
mouse
human
dog

[Number of Motifs] = 5
[Motif Mapping]
G      Gli1    Gli.txt motifmap_matrixscan -r 500 -b 3
O      OctSox  OctSox.txt     motifmap_matrixscan -r 500 -b 3
o      Oct     Oct.txt motifmap_matrixscan -r 500 -b 3
S      Sox     Sox.txt motifmap_matrixscan -r 500 -b 3
s      Sox2    Sox_2.txt      motifmap_matrixscan -r 500 -b 3
```

In this file,

“[Path of CisGenome] =” specifies the directory where executable CisGenome functions are installed.

“[Working Directory] =” specifies the directory where input and output files will be stored. The input files are the files generated by *malign_genome_blasthit*.

“[File Header] =” The title of input alignment files. This should be the same as the “[Export File] =” setting in the *malign_genome_blasthit* main parameter file.

“[Number of Clusters] =” specifies number of genes to be studied. This should be the same as the number of original genomic regions, or equivalently the number of “>” in the *malign_genome_blasthit* input file.

“[Number of Species] =” specifies the number of species in the alignments”

“[Species]” specifies species name in the alignment. The name should appear in the same order as they appear in the “[Target & Databases]” setting in the *malign_genome_blasthit* main parameter file.

“[Number of Motifs] =” specifies how many motifs one wants to map.

“[Motif Mapping]” specifies how to map each motif. If there are N motifs, “[Motif Mapping]” should be followed by N lines, each line in the following format:

```
MOTIF_SYMBOL[tab]MOTIF_NAME[tab]MOTIF_FILE[tab]Program and parameters
to map the motif.
```

Here, MOTIF_SYMBOL is a one letter character that will be displayed in the alignment to indicate the motif type. MOTIF_NAME is the name of the motif specified by users. MOTIF_FILE is the file name of the motif consensus or PWM. Program and parameters to map the motif can be chosen from motifmap_matrixscan and motifmap_consensusscan. The former can be used to map PWM, the latter can be used to map consensus. If motifmap_matrixscan is used, one also needs to specify “–r” (likelihood ratio cutoff) and “-b” (order of background model). If motifmap_consensusscan is used, one needs to specify “-mc” (maximal consensus mismatches allowed) and “-md” (maximal degenerate consensus mismatches allowed). Program name and parameters should be separated by space instead of tab.

After running malign_motifmap, one can check *[Ortholog cluster ID]_motif.aln for cross-species alignments annotated with TFBS.

9.7 Cross-species comparison Step IV – Searching for Modules according to the Annotated Alignments

```
[Usage]
malign_modulemap [Parameter file]
```

After cross-species alignment and TFBS annotations are obtained, one may want to get modules that contain certain types of TFBS. *malign_modulemap* can help to do this job. In order to run *malign_modulemap*, first prepare a parameter file (say *malign_modulemap_arg.txt*) as follows, then run the function as below:

```
> malign_modulemap malign_modulemap_arg.txt
```

A sample parameter file is shown as below:

```
[Working Directory] = .
[File Header] = target_3way
[Export File] = Gli-Oct_module.txt
[Number of Clusters] = 30
[Number of Species] = 3
[Number of Motifs] = 2
[Motif Criteria]
MID    MNAME   mouse   human    dog     Sum
G      Gli      1       0        0       2
O      Oct4     1       1        1       3
[Module Length] = 200
[Module Criteria]
MID     >=     =<
G      2      NA
O      1      3
```

In this file,

"[Working Directory] =" specifies the directory where input and output files will be stored. The input files are the files generated by *malign_motifmap*.

"[File Header] =" The title of input alignment files. This should be the same as the "[File Header] =" setting in the *malign_motifmap* main parameter file.

"[Export File] =" specifies the output file.

"[Number of Clusters] =" specifies number of genes to be studied. This should be the same as the number of original genomic regions, or equivalently the number of ">" in the *malign_genome_blasthit* input file.

"[Number of Species] =" specifies the number of species in the alignments"

"[Number of Motifs] =" specifies how many motifs are involved.

"[Motif Criteria]" specifies the criteria to filter TFBS. For each motif, one needs to set the one letter motif symbol, motif name, and whether a position needs to be called as TFBS in the given species (1: yes, 0: no) in order to be defined as a conserved TFBS. For example, the parameter file above says that a position can be called a conserved Gli site only if it is a TFBS in mouse (mouse>=1) and in at least one of human and dog (Sum>=2).

"[Module Length] =" specifies the length of a module.

"[Module Criteria]" specifies how many TFBS of each motif a region needs to contain in order to be a module. In the above example, a 200 bp long region will be called as a module if it contains >=2 Gli sites and 1~3 Oct4 sites.

Appendix.

A1. Commonly Used File Formats

A.1.1 SQ

A file in SQ format is a binary file. The file is used to store DNA sequences and must have a suffix .sq (e.g., chr1.sq, chrX.sq, etc.). In a *.sq file, each byte contains information for two nucleotides. Nucleotides are coded as below:

```
A: 0000
C: 0001
G: 0010
T: 0011
a: 0100
c: 0101
g: 0110
t: 0111
N: 1000
```

Here, “a”, “c”, “g”, “t” represent soft-masked repeat sequences; “A”, “C”, “G”, “T” represent non-repeat sequences; “N” represents all other sequences including gaps. Since each byte in a *.sq file codes two base pairs, storing a complete human genome (~3Gbp) needs ~1.5GB disk space. The genome is usually saved in files named as chr1.sq, chr2.sq, etc. The length of each *.sq file is approximately half of the length of the corresponding chromosome (e.g., the size of human chrX.sq is 77,412,132 bytes).

A.1.2 CS

A file in CS format is a binary file. The file is used to store various kinds of conservation scores and must have a suffix .cs (e.g., chr1.cs, chrX.cs, etc.). In a *.cs file, each byte corresponds to a single genomic position. Each byte contains a score that ranges from 0 to 255. The bigger the score, the more conserved the corresponding position is. Conservation scores for a complete genome are usually saved in files named as chr1.cs, chr2.cs, etc. ~3GB disk space is needed to store human conservation scores.

A.1.3 CDS

A file in CDS format is a binary file. The file is used to store protein coding region indicators and must have a suffix .cds (e.g., chr1.cds, chrX.cds, etc.). In a *.cds file, each byte corresponds to a single genomic position. Each byte contains a score that is either 0 or 1. If the score is 1, the corresponding position is located within a protein coding region; otherwise the score is 0. CDS indicators for a complete genome are usually saved in files named as chr1.cds, chr2.cds, etc. ~3GB disk space is needed to store human CDS indicators.

A.1.4 FASTA

A file in FASTA format is a text file. The file is used to store sequences. A FASTA file has the following format:

```
>seq0
GCTTAGCAACAGCTCACCAAAGTAGAGAGACCACCCAGGTAGGCAACCCCCGTGTGTGCA
TCCCAGGCTTGGGGGTGGGGGGGCGCTCGCTCAGCGCCAACCCTCTCGCATGCAATACTT
GTGTCACCAAGACAT
>seq1
TTTCAATGTGTCCTAACTGTTTGGAATAAATCTAAGGTTGTCCCTAGTTGTCATGGCATT
CAACCCTTTTCAATGGACTATCTCTTCATTCATTTCTGAATCCGTCCACAATATTAAGGA
AACCCCTTCATGTCGACGTTCCCCATTCCTCTTTCTCCTGCTTTTCTTTTCTCCTTTCTT
CCTCCCTTTTCCCCTTGCAAGAATTAGTATCTGGTTTGAAACGTG
```

A.1.5 BED

A file in BED format is a text file. Detailed description of BED format can be found at http://genome.ucsc.edu/goldenPath/help/customTrack.html. A BED file can be uploaded to UCSC Genome Browser and visualized as a customer track.

Below is a sample BED file that is commonly used in CisGenome:

```
browser position chr13:60894523-60943510
track name=ChipTarget description="ChIP-Chip tiling region"
chr2   74133959   74195987
chr12  53197523   53497991
chr10  127073216  127153216
chr10  127073187  127073215
chr10  127072750  127073048
```

A.1.6 COD_C

A file in COD_C format is a tab-delimited text file. The file is used to specify physical coordinates of genomic regions. The file should contain >=5 columns, and the first five columns are:

```
Sequence_ID[tab]chromosome[tab]start[tab]end[tab]strand
```

"Sequence_ID" is a unique ID (a number or a string) assigned to a region. The ID is defined by users.

"Chromosome" is the chromosome name, e.g., chr1, chr2, chrX, chrY, etc.

"Start" and "End" are integers that specify genomic coordinates of a region. Both "start" and "end" are zero-based, i.e., the first position in a chromosome is indexed by 0, the second position is indexed by 1, and so on.

"Strand" specifies the strand (+ or -) of a region in relative to the genome assembly.

Below is a sample file in COD_C format:

```
0   chr13  60949397   60950371   +
1   chr2   146644626  146645570  +
2   chr13  60951377   60952953   -
3   chr12  53347310   53348354   +
4   chrX   52789693   52789967   -
5   chr12  53340519   53341015   +
6   chr11  77915220   77915560   +
```

A.1.7 COD_N

A file in COD_N format is a tab-delimited text file. The file is used to specify physical coordinates of genomic regions. Similar to COD_C format, a COD_N file also contains >=5 columns, and the first five columns are:

```
Sequence_ID[tab]chromosome[tab]start[tab]end[tab]strand
```

However, unlike COD_C format, the “chromosome” (the 2nd column) in a COD_N file is indexed by integer numbers. In other words, in stead of using chr1, chr2, …, chrX, chrY to specify human chromosomes, a COD_N file uses 1, 2, …, 23, 24.

Below is a sample file in COD_N format:

```
0   13  60949397   60950371   +
1   2   146644626  146645570  +
2   13  60951377   60952953   -
3   12  53347310   53348354   +
4   20  52789693   52789967   -
5   12  53340519   53341015   +
6   11  77915220   77915560   +
```

A.1.8 COD_FA

A file in COD_FA format is a tab-delimited text file. The file is used to specify physical coordinates of target regions in sequences that are stored in a FASTA file. A COD_FA file contains >=4 columns, and the first four columns are:

```
Sequence_ID[tab]start[tab]end[tab]strand
```

"Sequence_ID" is a unique numerical ID assigned to the sequences in the FASTA file. The first sequence in the FASTA file is indexed by 0, the second sequence is indexed by 1, and so on.

"Start" and "End" are integers that specify coordinates of a region. Both "start" and "end" are zero-based, i.e., the first position in a sequence is indexed by 0, the second position is indexed by 1, and so on.

"Strand" specifies the strand (+ or -) of a region in relative to the FASTA sequence.

Below is a sample file in COD_FA format:

```
0   3187   3199   -
0   3903   3915   +
1   3999   4011   +
1   7666   7678   +
2   8252   8264   +
3   9026   9038   +
3   9141   9153   -
3   9603   9615   +
```

In this sample file, the line "`0 3187    3199    -`" specifies a region in the first sequence that starts from the 3188th base pair and ends at 3200th base pair; the line "`3  9141    9153    -`" specifies a region in the fourth sequence that starts from the 9142th base pair and ends at 9154th base pair, and so on.

A.1.9 MOTIF_CONS

A file in MOTIF_CONS format is a text file. The file is used to specify consensus sequence motifs. Each file (say Gli_cons.txt) can specify a single consensus motif in the following format:

```
TGGGT[A]GGTC[G,T]
```

In the above example, "TGGGT[A]GGTC[G,T]" can be viewed as a motif as below:

```
TGGGTGGTC
----A---G
--------T
```

In other words, the 5th position of the motif could be either "T" or "A", and the 9th position could be "C", "G" or "T".

Capital letters "A", "C", "G", "T", "[", "]" and "," are the only characters that can be used in a MOTIF_CONS file. Their combinations can be used to specify a wide range of nucleotide preference, e.g., "A[C,G,T]" can be used to represent "N".

Two relevant concepts when applying a motif in MOTIF_CONS format is the *consensus sequence* and *degenerate consensus sequence*. In the above example, "TGGGTGGTC" is defined as the *consensus sequence*, and "TGGGT[A]GGTC[G,T]" is defined as the *degenerate consensus sequence*. A binding site "TGGGAGGTA" has 2 mismatches to the consensus (or 2 *consensus mismatches*), and 1 mismatch to the degenerate consensus (or 1 *degenerate mismatch*). The 2 consensus mismatches are TGGG**A**GGT**A**, and the 1 degenerate mismatch is TGGGAGGT**A**.

A.1.10 MOTIF_MAT

A file in MOTIF_MAT format is a text file. The file is used to specify motif position specific weight matrices (PWM). Each file (say Gli_mat.txt) can specify a single PWM in the following format:

```
0.1  1.1   0.1   14.1
0.1  0.1  12.1    3.1
0.1  0.1  15.1    0.1
0.1  0.1  15.1    0.1
2.1  0.1   1.1   12.1
0.1  0.1  14.1    1.1
0.1  0.1  13.1    2.1
1.1  1.1   1.1   12.1
0.1 12.1   2.1    1.1
```

Each line in the file corresponds to a position in the motif, and the four columns count A, C, G, and T respectively. In the above example, the motif has a consensus TGGGTGGTC.

Usually, users need to add a small number (say 0.1) to each count to make sure that no counts <= 0. This small number is added to avoid log(0) when applying various motif mapping functions. Although many CisGenome motif mapping functions will automatically add 0.001 to each count in the PWM, we strongly recommend that users add this pseudo-counts themselves to control their potential effects in motif mapping.

A.1.11 MOTIF_SITE

A file in MOTIF_SITE format is a tab-delimited text file. The file is used to store transcription factor binding sites. Each line in the file contains information for a single TFBS. The information is provided in the following format:

Sequence_ID[tab]chromosome[tab]start[tab]end[tab]strand[tab]site_score[tab]site

"Sequence_ID" is an ID (a number or a string) assigned to the genomic region or the FASTA sequence where the TFBS has been found. If the file is generated by *motifmap_consensusscan_genome* or *motifmap_matrixscan_genome*, the ID is transferred form the first column of the input coordinates file (a file in COD_C format). If the file is generated by *motifmap_consensusscan* or *motifmap_matrixscan*, the ID is defined by lines starting with ">" in the input sequences (a FASTA file). If the file is generated by users, users can define the ID by themselves.

"Chromosome" is the chromosome name. If the file is generated by *motifmap_consensusscan_genome* or *motifmap_matrixscan_genome*, the chromosome is reported as chr1, chr2, chrX, chrY, etc. If the file is generated by *motifmap_consensusscan* or *motifmap_matrixscan*, the chromosome is defined as a number that indexes the input FASTA sequences. The index is 0-based, i.e. the first sequence is indexed as 0, the second sequence indexed by 1, and so on.

"Start" and "End" are integers that specify coordinates of a TFBS. Both "start" and "end" are zero-based, i.e., the first position in a chromosome (or FASTA sequence) is indexed by 0, the second position is indexed by 1, and so on.

"Strand" specifies the strand (+ or -) of a TFBS in relative to the genomic regions or FASTA sequences.

"Site_score" is the score attached to the site.

"Site" is the sequence of the site.

Below are two sample files in MOTIF_SITE format:

```
0   chr13  60949425   60949433   -   4.480395   TGGGTGGTC
1   chr2   146645187  146645195  +   4.525546   TGGGTGGTC
2   chr13  60951437   60951445   +   3.048943   TGGGTGTTA
2   chr13  60952275   60952283   +   4.490870   TGGGTGGTC
```

```
0   0   28  36  -   4.148830   TGGGTGGTC
1   1   561 569 +   4.178913   TGGGTGGTC
2   2   60  68  +   3.088845   TGGGTGTTA
2   2   898 906 +   4.311735   TGGGTGGTC
```

Hint: A file in MOTIF_SITE format sometimes can also be viewed as a file in COD_C or COD_N format, therefore it may be used directly in sequence retrieval functions such as *genome_getseq*, *genome_getseqcs*, etc.

A.1.12 MOTIF_MASK

A file in MOTIF_MASK format is a text file. The file is used to specify specific positions in a motif that need to be processed. The file has a single line in the format:

```
1 1 1 0 0 1 1 1
```

Positions labeled by "1" will be processed by programs such as *motifmap_filter_genome*, whereas positions labeled by "0" will be ignored in the analysis. For example, if the motif is CCCATGGG, then by applying the above mask, the underscored positions will be analyzed: CCCATGGG.

A.1.13 MOTIF_LIST

A file in MOTIF_LIST format is a tab-delimited text file. The file is used to specify a list of motifs and their corresponding likelihood ratio cutoffs used for *motifmap_matrixscan_genome_summary*. Each line in the file corresponds to a single motif, and the line has the following format:

```
PWM_FILE[tab]RATIO_CUTOFF
```

Here, PWM_FILE is the path of the file that specifies motif PWM. The PWM should be stored in MOTIF_MAT format. RATIO_CUTOFF is a number that specifies the likelihood ratio cutoff used to declare TFBS when mapping the motif to genomic regions. Below is a sample MOTIF_LIST file:

```
Gli_mat.txt	500
Motif1.txt	500
Motif2.txt	500
Motif3.txt	500
```

A.1.14 MOTIF_SUMMARY

A file in MOTIF_SUMMARY format is a tab-delimited text file. The file is used to summarize the relative enrichment level of a list of motifs in target genomic regions as compared to control regions. Each line in the file corresponds to a single motif, and the line has the following format:

MOTIF_ID[tab]MOTIF[tab]n_{1B}[tab]n_{2B}[tab]n_{1C}[tab]n_{2C}[tab]r_1[tab]n_{3B}[tab]n_{4B}[tab] n_{3C}[tab] n_{4C}[tab]r_2[tab]r_3[tab]

Here, MOTIF_ID is a numerical index attached to each motif. MOTIF is the file that specifies the motif PWM. The remaining columns are summary statistics discussed in section 8.8.

Below is a sample MOTIF_SUMMARY file:

```
1   motif1.txt 165 26361   16322   5606144    2.149874   71  12542   4378
        1223995    1.582688   3.448935
2   motif2.txt 102 26361   10326   5606144    2.100731   26  12542   1320
        1223995    1.922261   4.188917
3   Motif3.txt 40  26336   2347    5605719    3.627675   26  12513   839
        1222435    3.027441   6.596186
```

A.1.15 MOTIF_TIERENRICH

A file in MOTIF_TIERENRICH format is a tab-delimited text file. The file is used to summarize the relative enrichment level of a motif in target genomic regions as compared to control regions. Target regions are ranked and grouped into tiers. Each line in the file summarizes the relative enrichment level of a single tier. The last line contains summary statistics for control regions. Typically, a line has the following format:

Tier[tab]n_{1B}[tab]n_{2B}[tab]r_1[tab]n_{3B}[tab]n_{4B}[tab]r_2[tab]r_3[tab]

The last line has the following format:

Control[tab]n_{1C}[tab]n_{2C}[tab]r_1[tab]n_{3C}[tab]n_{4C}[tab]r_2[tab]r_3[tab]

The meaning of the summary statistics discussed in section 8.8. A sample file in MOTIF_TIERENRICH format is given as below:

```
1~10        19     10820   4.033939   13   5328  5.494526   14.106073
11~20       16      9779   3.758621   12   4838  5.585556   14.407108
21~30        6      6280   2.194798    4   2720  3.311633    7.478085
31~40        3      4959   1.389729    1   1594  1.412742    2.367532
41~50        3      4919   1.401030    1   1826  1.233248    2.386784
51~60        2      4159   1.104699    1   1241  1.814593    2.822936
61~65        1      1877   1.223880    0    779  0.000000    0.000000
Control   1707   3921358   1.000000  334 752138  1.000000    1.000000
```

A.1.16 REFGENE_UCSC

A file in REFGENE_UCSC format is a tab-delimited text file. Detailed description of this format can be found at http://genome.ucsc.edu/goldenPath/gbdDescriptions.html#GenePredictions.

Each line in the file contains following gene information:

Name[tab]chromosome[tab]strand[tab]transcription_start[tab]transcription_end[tab]coding_region_start[tab]coding_region_end[tab]exon_count[tab]exon_starts[tab]exon_ends[tab].

Here, chromosome is in a format such as chr1, chrX, chrY, etc. Several sample lines are shown below:

```
NM_198943  chr1   -   4268   14754  4558   14749  10
    4268,4832,5658,6469,6719,7095,7468,7777,8130,14600,
    4692,4901,5810,6631,6918,7231,7605,7924,8242,14754,
NM_182905  chr1   -   4558   7173   4558   7173   6
    4558,4832,5658,6469,6719,7095,  4692,4901,5810,6631,6918,7173,
NM_024796  chr1   -   801449 802749 801942 802434 1   801449,
    802749,
```

A.1.17 REFGENE_CISGENOME

A file in REFGENE_CISGENOME format is a tab-delimited text file. It has almost the same format as REFGENE_UCSC. The only differences are: (1) in REFGENE_CISGENOME, chromosomes (column 2) are represented by numbers instead of strings, and (2) genes are sorted according to their genomic coordinates.

Each line in the file contains following gene information:

Name[tab]chromosome[tab]strand[tab]transcription_start[tab]transcription_end[tab]coding_region_start[tab]coding_region_end[tab]exon_count[tab]exon_starts[tab]exon_ends[tab].

Several sample lines are shown below:

```
NM_198943  1   -   4268   14753  4558   14748  10
    4268,4832,5658,6469,6719,7095,7468,7777,8130,14600,
    4691,4900,5809,6630,6917,7230,7604,7923,8241,14753,
NM_182905  1   -   4558   7172   4558   7172   6
    4558,4832,5658,6469,6719,7095,  4691,4900,5809,6630,6917,7172,
NM_001005484  1   +   58953  59870  58953  59870  1   58953, 59870,
NM_001005277  1   +   407521 408459 407521 408459 1   407521,
    408459,
```

A.1.18 REFFLAT_UCSC

A file in REFFLAT_UCSC format is a tab-delimited text file. Detailed description of this format can be found at http://genome.ucsc.edu/goldenPath/gbdDescriptions.html#RefFlat.

Each line in the file contains following gene information:

Gene_symbol[tab]name[tab]chromosome[tab]strand[tab]transcription_start[tab]transcription_end[tab]coding_region_start[tab]coding_region_end[tab]exon_count[tab]exon_starts[tab]exon_ends[tab].

Here, chromosome is in a format such as chr1, chrX, chrY, etc. Several sample lines are shown below:

```
NOC2L  NM_015658  chr1   -   919738 934774 920216 934763 19
    919738,920579,921040,921695,921924,923653,924012,926649,927522,92
   7934,928697,929304,929526,931445,931617,932416,932621,934451,93473
   7,
    920323,920669,921176,921809,922068,923755,924126,926761,927662,92
   8123,928811,929415,929605,931536,931738,932548,932796,934604,93477
   4,
KLHL17 NM_198317  chr1   +   936109 941162 936216 940638 12
    936109,936815,937151,937348,937877,938226,938631,938859,939442,93
   9629,939871,940409,
    936323,937075,937273,937570,937994,938440,938776,939027,939531,93
   9703,940053,941162,
PLEKHN1    NM_032129  chr1   +   941943 950549 941978 950022 16
    941943,942150,945723,945967,946132,946325,946559,946770,947521,94
   7734,948307,948632,948946,949279,949762,949888,
    942061,942250,945870,946048,946205,946453,946655,946851,947597,94
   7871,948457,948773,949087,949498,949811,950549,
```

A.1.19 REFFLAT_CISGENOME

A file in REFFLAT_CISGENOME format is a tab-delimited text file. It has almost the same format as REFFLAT _UCSC. The only differences are: (1) in REFFLAT_CISGENOME, chromosomes (column 3) are represented by numbers instead of strings, and (2) genes are sorted according to their genomic coordinates.

Each line in the file contains following gene information:

Gene_symbol[tab]name[tab]chromosome[tab]strand[tab]transcription_start[tab]transcription_end[tab]coding_region_start[tab]coding_region_end[tab]exon_count[tab]exon_starts[tab]exon_ends[tab].

Several sample lines are shown below:

```
NOC2L  NM_015658  1   -   919738 934773 920216 934762 19
    919738,920579,921040,921695,921924,923653,924012,926649,927522,92
   7934,928697,929304,929526,931445,931617,932416,932621,934451,93473
   7,
    920322,920668,921175,921808,922067,923754,924125,926760,927661,92
   8122,928810,929414,929604,931535,931737,932547,932795,934603,93477
   3,
KLHL17 NM_198317  1   +   936109 941161 936216 940637 12
    936109,936815,937151,937348,937877,938226,938631,938859,939442,93
   9629,939871,940409,
    936322,937074,937272,937569,937993,938439,938775,939026,939530,93
   9702,940052,941161,
PLEKHN1    NM_032129  1   +   941943 950548 941978 950021 16
    941943,942150,945723,945967,946132,946325,946559,946770,947521,94
   7734,948307,948632,948946,949279,949762,949888,
    942060,942249,945869,946047,946204,946452,946654,946850,947596,94
   7870,948456,948772,949086,949497,949810,950548,
```

A.1.20 REFGENE_CISGENOME_FLEXIBLE

A file in REFGENE_CISGENOME_FLEXIBLE format is a tab-delimited text file. It can be viewed as an extension of REFGENE_CISGENOME format. Both REFGENE_CISGENOME and REFFLAT_CISGENOME format are special cases of REFGENE_CISGENOME_FLEXIBLE format.

Each line in a REFGENE_CISGENOME_FLEXIBLE file has the following format:

Info1[tab]Info2[tab]…[tab]InfoN[tab]Name[tab]chromosome[tab]strand[tab]transcription_start[tab]transcription_end[tab]coding_region_start[tab]coding_region_end[tab]exon_count[tab]exon_starts[tab]exon_ends[tab].

In other words, *N* columns of various types of information are followed by a gene structure annotation, and the gene structure annotation is in REFGENE_CISGENOME format. According to this definition, REFGENE_CISGENOME is a special case of REFGENE_CISGENOME_FLEXIBLE format where *N*=0; and REFFLAT_CISGENOME format is a special case where *N*=1. Another example which is commonly used is shown as below:

```
region1	chr2	32395049	32400049	-	4933440H19Rik	NM_194335	2	+	32382619	32387638	32384601	32387431	2	32382619,32386959,	32385111,32387638,
region2	chr2	61504133	61509133	+	Tank	NM_011529	2	+	61433985	61509511	61434041	61508955	8	61433985,61469051,61482303,61483121,61499115,61499707,61505138,61508779,	61434093,61469198,61482414,61483239,61499185,61499822,61505688,61509511,
region3	chr2	129329174	129334174	+	Stk35	NM_183262	2	+	129314605	129341840	129315476	129325053	4	129314605,129315257,129324341,129341627,	129314699,129315857,129325090,129341840,
```

Here, the first five columns specifies the coordinates of a genomic region (similar to COD_C format), the sixth column is a gene name (used to annotate the region), and the remaining columns are the gene structure in REFGENE_CISGENOME format. This file can be generated by *refgene_getnearestgene* which is used to annotate specified genomic regions, and may be used as the input file for *refflex_getmultiortholog* to get ortholog genes in other species.

A.1.21 REFGENE_ORTHOLOG

A file in REFGENE_ORTHOLOG format is a text file. The file is usually used to save ortholog information for a group of genes. A gene and its orthologs in other species are called a ortholog group. Each ortholog group is saved in the file in the following format:

>Group_ID
Information of the original gene (in REFGENE_CISGENOME_FLXIBLE format)
Species1[tab]M[tab]2*O[tab]Information of the ortholog gene in species 1 (in REFGENE_CISGENOME format)
Species2[tab]M[tab]2*O[tab]Information of the ortholog gene in species 2 (in REFGENE_CISGENOME format)
…
SpeciesN[tab]M[tab]2*O[tab]Information of the ortholog gene in species N (in REFGENE_CISGENOME format)

Here, "M" and "O" are defined in section 4.7. Below is an example:

```
>0
NM_008625	chr2	14173834	14178834	+	Mrc1
   NM_008625	2	+	14155686	14258061
   14155785	14257447	30
   14155686,14164398,14170390,14175079,14183276,14187461,14192622,141
   96411,14197536,14202559,14206108,14215259,14218791,14219718,142210
   72,14230383,14231588,14232736,14234159,14234951,14236240,14241506,
   14243114,14245373,14248045,14251500,14254059,14254692,14256155,142
   57194,
   14155845,14164799,14170563,14175243,14183389,14187607,14192807,141
   96568,14197646,14202674,14206256,14215458,14218918,14219805,142212
   16,14230424,14231751,14232803,14234259,14235093,14236354,14241672,
   14243216,14245605,14248210,14251649,14254172,14254856,14256196,142
   58061,
mouse	5	12	NM_008625	2	+	14155686
   14258061	14155785	14257447	30
   14155686,14164398,14170390,14175079,14183276,14187461,14192622,141
   96411,14197536,14202559,14206108,14215259,14218791,14219718,142210
   72,14230383,14231588,14232736,14234159,14234951,14236240,14241506,
   14243114,14245373,14248045,14251500,14254059,14254692,14256155,142
   57194,
   14155845,14164799,14170563,14175243,14183389,14187607,14192807,141
   96568,14197646,14202674,14206256,14215458,14218918,14219805,142212
   16,14230424,14231751,14232803,14234259,14235093,14236354,14241672,
   14243216,14245605,14248210,14251649,14254172,14254856,14256196,142
   58061,
human	4	6	NM_008625	10	+	18138450
```

```
    18239602              18138460              18239399              33
    18138450,18141886,18152048,18156500,18162635,18169627,18174223,181
    78514,18181901,18185176,18190321,18192468,18195496,18199180,181997
    86,18200916,18202718,18204024,18208656,18209459,18210001,18214203,
    18223144,18224326,18226967,18227177,18229722,18230907,18235820,182
    36468,18239211,18239496,18239582,
    18138512,18141892,18152450,18156669,18162797,18169740,18174369,181
    78698,18182058,18185244,18190436,18192616,18195695,18199309,181998
    69,18201060,18202758,18204192,18208755,18209535,18210115,18214369,
    18223239,18224331,18227146,18227200,18229887,18231056,18235933,182
    36632,18239400,18239571,18239602,
dog      4        6        NM_008625        2        -        19668039
    19750134              19668627              19750134              32
    19668039,19668387,19668458,19668623,19671171,19671868,19678493,196
    79478,19681414,19684650,19688062,19692192,19692689,19693703,196967
    56,19698121,19699690,19702173,19702411,19702745,19705839,19708163,
    19710848,19716640,19717871,19726880,19730029,19733357,19741053,197
    45645,19745809,19749732,
    19668097,19668400,19668513,19668814,19671334,19671981,19678642,196
    79643,19681647,19684745,19688228,19692306,19692831,19693802,196969
    24,19698161,19699834,19702255,19702414,19702873,19706038,19708311,
    19710963,19716705,19718028,19727063,19730175,19733470,19741217,197
    45648,19745977,19750134,
```

A.1.22 GENOME_SEQUENCE_SETTING

A file in GENOME_SEQUENCE_SETTING format is usually used to tell programs where to find sequences, conservation scores and annotations for multiple species. A sample file is shown as below:

```
[Strand Type] = assemblywise
[Conservation Type] = cs
[Species Number] = 3
[Species Name] = mouse
[Species Genome] = /data/genomes/mouse/mm6/
[Species Conservation] = /data/genomes/mouse/mm6/conservation/genomelab/cs/
[Species Annotation] = /data/genomes/mouse/mm6/annotation/
[Species Name] = human
[Species Genome] = /data/genomes/human/hg17/
[Species Conservation] = /data/genomes/human/hg17/conservation/phastcons/
[Species Annotation] = /data/genomes/human/hg17/annotation/
[Species Name] = dog
[Species Genome] = /data/genomes/dog/canFam1/
[Species Conservation] = NULL
[Species Annotation] = /data/genomes/dog/canFam1/annotation/
```

Here, “[Strand Type] =” specifies how the sequences will be extracted (*assemblywise* or *genewise*, see discussions in section 3.3). We recommend *assemblywise*.

“[Conservation Type] =” specifies the suffix of genome conservation score files. Usually one needs to set *cs* here.

“[Species Number] =” specifies how many species there are. If there are N species, then this line should be followed by N sections including “[Species Name]”, “[Species Genome]”, “[Species Conservation]” and “[Species Annotation]”.

“[Species Name] =” specifies the species name, e.g., human, mouse, dog, cow, etc.

“[Species Genome] =” specifies the directory where the genome sequences (*.sq files) are stored.

“[Species Conservation] =” specifies the directory where the genome conservation scores (*.cs files) are stored. If no conservation scores are available, set “NULL” here.

“[Species Annotation] =” specifies the directory where the genome annotation files (e.g., refGene_sorted.txt, refFlat_sorted.txt) are stored.

A.1.23 GENOME_ANNOTATION_SETTING

A file in GENOME_ANNOTATION_SETTING format can be used to tell programs where to find gene annotations. This file is usually used in identifying orthologs. A sample file is shown as below:

```
[Species Number] = 3
[Reference Species] = mouse
[RefId From RefGeneMap(0) or RefGene(1)] = 0
[Reference RefGeneMap] = /data/genomes/mouse/mm6/annotation/refGene_sorted.txt
[Reference RefGene] = NULL
[Map Species Name] = human
[Map Species RefGeneMap] = /data/genomes/human/hg17/annotation/
   mouseRefGene_sorted.txt
[Map Species RefGene] = /data/genomes/human/hg17/annotation/refGene_sorted.txt
[Map Species Name] = dog
[Map Species RefGeneMap] = /data/genomes/dog/canFam1/annotation/
   mouseRefGene_sorted.txt
[Map Species RefGene] = NULL
```

Here, "[Species Number] =" specifies how many species there are. If there are N species, then this line should be followed by 1 reference section and N-1 map sections. The reference section specifies information for the reference species. A list of genes from reference species often is the starting point for identifying orthologs in other species. The map sections contain information for the other species from which orthologs need to be identified.

In the reference section,

"[Reference Species] =" specifies the reference species name, e.g., human, mouse, dog, cow, etc.

"[RefId From RefGeneMap(0) or RefGene(1)] =": Before ortholog identification, one needs to provide a list of RefSeq ID's. If the RefSeq ID's origin from the reference species, then set 1 here. If the RefSeq ID's origin from the link species (refer to section 4.7), then set 0 here.

"[Reference RefGeneMap] =" specifies the file that contains refGene annotations for all RNAs that origin from the link species. The annotations are derived from the alignments of link species RefSeqs to the reference species genome.

"[Reference RefGene] =" specifies the file that contains refGene annotations for all RNAs that origin from the reference species. The annotations are derived from the alignments of reference species RefSeqs to the reference species genome. If such a file is not available, specify "NULL" here.

In each map section,

"[Map Species Name] =" specifies the map species name in which one wants to search for orthologs.

“[Map Species RefGeneMap] =” specifies the file that contains refGene annotations for all RNAs that origin from the link species. The annotations are derived from the alignments of link species RefSeqs to the genome of map species.

“[Map Species RefGene] =” specifies the file that contains refGene annotations for all RNAs that origin from the map species. The annotations are derived from the alignments of map species RefSeqs to the genome of map species. If specified, RefSeq IDs returned in the final results of various programs will have a map species origin. If such a file is not available, specify “NULL” here, correspondingly RefSeq IDs returned in the final results will origin from the link species.

A.1.24 EXPRESSION_DATA

A file in EXPRESSION_DATA format is a tab-delimited text file. The file is usually used to store microarray expression data. The file has the following format:

1st row: array id
2nd row and after: data
1st col: probeset id
2nd col and after: data

A sample file is shown below.

```
Probe_set	E8.75.SMO.a	E8.75.SMO.b	E8.75.WT.a	E8.75.WT.b
100001_at	132.724	112.445	128.478	154.888
100002_at	161.825	163.304	210.121	159.003
```

A.1.25 EXPRESSION_ANNOTATION

A file in EXPRESSION_ANNOTATION format is a tab-delimited text file. The file is usually used to provide annotations for microarray probes/probesets. The file has following format:

1st row: field name
2nd row and after: annotations
1st col: probeset id
2nd col: gene identifier
3rd col: Entrez gene id
4th col and after: any user provided tab-delimited annotations

A sample file is shown below:

```
Probe.Set.Name    Identifier LocusLink  Name
1417246_at    NM_007376  11287  pregnancy zone protein
1421666_a_at  NM_009591  11298  arylalkylamine
```

A.1.26 TILEMAP_DATA

A file in TILEMAP_DATA is a tab-delimited text file. Raw tiling array data are usually organized in this format. In the file:

1st row: 'chromosome', 'position', array ids
2nd row and after: each row corresponds to a probe; probes should be arranged in the same order as they appear in the genome.

1st col(column): chromosome name
2nd col: genomic coordinate of the probe
3rd col and after: probe intensity data. Each column corresponds to an array.

Below is a sample file:

```
chromosome position   IP1 IP2 CT1 CT2
chr21  9928660    4.8145015e-002    -2.6689525e-001   -3.4340357e-001
    -3.6077572e-001
chr21  9928688    2.8436127e-001    -2.3055024e-001   -2.0545930e-001
    2.3260081e-001
chr21  9928724    -2.4290403e-002   1.6691128e-002    5.7574268e-002
    -7.5443205e-002
```

A.1.27 TILEMAP_COMPINFO

This is a text file used to specify how to compare different experimental conditions in TileMap. Below is a sample file:

```
##############################
# TileMap Comparison Info    #
##############################

##############################
# Basic Info              #
##############################
[Array number] = 18
[Group number] = 3
[Group ID]
1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3

##############################
# Patterns of Interest      #
##############################
[Comparisons]
(1>2) & (1>3)

##############################
# Preprocessing            #
##############################
[Truncation lower bound] = -100000000.0
[Take log2 before calculation?] (1:yes; 0:no) = 0

##############################
# Simulation Setup         #
##############################
[Monte Carlo draws for posterior prob.] = 1000

##############################
# Common Variance Groups    #
##############################
[Common variance groups] = 1
1 2 3

##############################
# Permutation Setup         #
##############################
[Number of permutations] = 10
[Exchangeable groups] = 1
```

```
1 2 3
```

The meanings of the parameter settings are explained below.

(A) In "Basic Info" section, one needs to provide general information about the tiling array experiment.

[Array number]: the number of arrays to be analyzed.
[Group number]: the number of experimental conditions.
[Group ID]: numerical group ID for individual arrays. The ID's are arranged in the same order as the order arrays (columns) appear in the raw data file. IDs range from 1 to [Array number]. Negative integers can be used if one wish to ignore a specific column in the raw data file.

For example, if three mice strain "wt", "mt1" and "mt2" were profiled, each with 6 replicates. The 18 arrays are arranged in the raw data file as:
{chromosome position wt wt wt mt1 mt1 mt1 mt2 mt2 mt2 wt wt wt mt1 mt1 mt1 mt2 mt2 mt2}
then

[Array number] = 18
[Group number] = 3
[Group ID]
1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3

If one wishes to exclude the last nine arrays from the analysis, one can set
[Array number] = 9
[Group number] = 3
[Group ID]
1 1 1 2 2 2 3 3 3 -1 -1 -1 -2 -2 -2 -3 -3 -3

(B) In "Patterns of Interest" section, one sets the transcriptional or protein binding patterns of interest.

For example, if one wants to select regions that show "mt1<wt<mt2", the criteria can be set as [Comparisons]
(2<1) & (1<3)

If one wants to select regions that show "mt1<wt OR wt<mt2", the criteria can be set as
[Comparisons]
(2<1) | (1<3)

Currently, we only support the following operations:
< -- (less than)
> -- (greater than)
& -- (and)
| -- (or)
() -- (to specify operational priorities)

(C) In "Preprocessing" section, one specifies how to truncate low expression values and whether log-transformation should be taken before analysis.

For example, if one wishes to truncate all intensities that are less than 2 and set them to be 2, and wishes to take log2 transformation after the truncation, one can set
[Truncation lower bound] = 2.0
[Take log2 before calculation?] (1:yes; 0:no) = 1

If one has already done truncations and log-transformations in tilemap_importaffy or tilemap_norm, one doesn't need to do preprocessing again. In this case, one can set
[Truncation lower bound] = -1000000000000.0
[Take log2 before calculation?] (1:yes; 0:no) = 0

(D) In "Simulation Setup" section, one specifies how many Monte Carlo draws should be made to estimate the posterior probability that a probe satisfies the pattern of interest. If the comparisons specified in "Patterns of Interest" section is a two sample comparison (e.g. "1<2"), there is no need to do Monte Carlo, therefore one can set
[Monte Carlo draws for posterior prob.] = 0

If the "Patterns of Interest" involves a multiple sample comparison (e.g. "(1<2) & (1<3)"), one needs to specify a positive number, for example
[Monte Carlo draws for posterior prob.] = 1000

(E) In "Common Variance Groups", one needs to specify which experimental conditions are assumed to have common variance. The variance shrinking will be based on this setting. For example, if there are six conditions, and one assumes that for each probe, condition 1,2,3 have common within-condition variance, condition 4,5,6 have common within-condition variance, but the within-condition variance for 1,2,3 is different from within-condition variance for 4,5,6, then there are 2 common variance groups, and one can set:

variance group = 2

1 2 3
4 5 6

Each line below "variance group" tag corresponds to a common variance group, which contains all the conditions that are assumed to have common variance. The variance shrinking will be done within each variance group.

If you are not sure how to set variance group, you can assume that all conditions have the same variance. For example, you can set

variance group = 1
1 2 3 4 5 6

Setting variance group appropriately can increase the sensitivity of the analysis, especially when the number of replicate arrays are small.

(F) In "Permutation Setup", one specifies how to do permutations if one chooses to use permutation test to estimate local false discovery rate in MA.

[Number of permutations]: how many times to permute group labels.
[Exchangeable groups]: conditions that can be permuted.

For example, if one set
[Number of permutations] = 10
[Exchangeable groups] = 2
1 2 3
4 5 6

then the labels in "Group ID" will be permuted 10 times for computing FDR. The labels are permuted according to "Exchangeable groups". Arrays labeled by 1,2 and 3 will only be permuted with arrays labeled by 1, 2 and 3. Similarly, arrays labeled by 4, 5 or 6 will only be permuted with arrays labeled by 4, 5 and 6. No permutations will be done between "1, 2, 3" and "4, 5, 6". In other words, the FDR computed is a FDR for a null hypothesis H0: "1=2=3, 4=5=6".

If one wish to compute a FDR for H0: "1=2=3=4=5=6", one can set
[Number of permutations] = 10
[Exchangeable groups] = 1
1 2 3 4 5 6

If one does not want to use permutation test to compute FDR, one can set
[Number of permutations] = 10
[Exchangeable groups] = 1

1 2 3 4 5 6

Depending on the data size, permutation test may require a long time. Moreover, it is hard to estimate FDR for H0: "not {1<2<3}" using permutation test.

REFERENCES

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262, 208-214.

Liu, J. S. (1994). The collapsed Gibbs sampler with applications to a gene regulation problem. *Journal of the American Statistical Association* 89, 958-966.

Liu, J. S., Neuwald, A. F., and Lawrence, C. E. (1995). Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *Journal of the American Statistical Association* 90, 1156-1170.