HMMR Quick Start:

**Section 1: Creating required files**

1. The first file that is necessary is a **sorted BAM file** containing the pair-end ATAC-seq reads. Typically, an alignment algorithm (such as bowtie/bowtie2) will output a SAM file as default. The first step is to convert this SAM file into a BAM file. This is best accomplished using samtools. Samtools requires a SAM file and a FASTA file to convert into a BAM file. Use the samtools “view” command as follows:

**$ samtools view –bT hgXX.fa XXX.sam > XXX.bam**

The next step is to sort the BAM file. This can also be accomplished using samtools. Use the samtools “sort” command as follows:

**$ samtools sort XXX.bam XXX.sorted.bam**

It is possible to remove duplicates or low Mapping quality reads before inputting into HMMR, although HMMR will perform these functions by default. By default HMMR will remove duplicate reads (and this function is currently hard-coded and cannot be turned off). HMMR also removes those reads whose MapQ (mapping quality scores) are below 30. This value is changeable at runtime (using the –q or –-minmapq option)

If you choose to merge replicates prior to running HMMR, this can also be accomplished using samtools. Use the samtools “merge” command as follows:

**$ samtools merge –n MergedFile.bam XXX\_rep1.sorted.bam XXX\_rep2.sorted.bam … XXX\_repN.sorted.bam**

If you do choose to merge replicates, you will have to re-sort the resulting merged file again before proceeding.

1. The next required file that HMMR needs is an **index file** for the sorted BAM file. This file can also be easily created with samtools. Use the samtools “index” command as follows:

**$ samtools index XXX.sorted.bam XXX.sorted.bam.bai**

1. The third required file is a **genome annotation file.** This file can be downloaded from numerous online sources, including UCSC genome browser’s website. This file is a two column, tab-delimited file containing chromosome name and size in the following format:

<chromName><TAB><chromSIZE>

It is possible to retrieve this file using UCSC Genome Browser’s MySQL database. To retrieve the file, for H. sapiens, use the following command:

**$ mysql –-user=genome --host=genome-mysql.cse.ucsc.edu –A –e \ “select chrom, size from hg19.chromInfo” > hg19.genome**

1. The final required file that is necessary to run HMMR is a **BigWig file** containing the genome-wide coverage of ATAC-seq reads. This can be created using multiple programs. One such method is to use bedtools. Use the bedtools “bamtobed” command as follows:

**$ bedtools bamtobed XXX.sorted.bam > XXX.sorted.bed**

The next step would be to use the BED file to create a genome-wide BedGraph coverage file. This could also be accomplished using bedtools. You would use the bedtools “genomecov” command as follows:

**$ bedtools genomecov –i XXX.sorted.bed –g hg19.genome –bga > XXX.sorted.bg**

It should be noted that bedtools bamtobed has not performed well in our experience. Many paired end reads are reported to be improperly paired, while other analysis shows that this isn’t the case. We have developed in-house programs for BAM to BED conversion, which will be available soon. However, we generally use MACS2 to accomplish this process, as MACS2 is quick and easy to use. MACS2 has the option (-B) to output a genome-wide BedGraph file. You can generate the file using the MACS2 command as follows:

**$ macs2 callpeak –t XXX.sorted.bam –f BAMPE –n XXX –B**

The resulting **XXX\_treat\_pileup.bg** represents the BedGraph file. The BedGraph file, however it is generated, then must be converted into a BigWig file. We use the UCSC program **bedGraphToBigWig** to accomplish this. The command is as follows:

**$ bedGraphToBigWig XXX.bg hg19.genome XXX.bw**

You should replace **XXX.bg** with the name of the BedGraph file previously created (such as **XXX.sorted.bg** OR **XXX\_treat\_pileup.bg**).

**Section 2: Overview of Options**

**Required Parameters:**

-b , --bam <BAM> This is the sorted BAM file created as described in section

1.1.

-i , --index <BAI> This is the index file for the sorted BAM file created as

described in section 1.2.

-g , --genome <GenomeFile> This is the genome file created or downloaded

as described in section 1.3

-w , --wig <BigWig> This is the genome-wide BigWig file created as

described in section 1.4

**Optional Parameters:**

-m , --means <Double> Comma separated list of initial mean values for the

fragment distribution, used to create the signal tracks. The default values

are 50,200,400,600. It is recommended to change these values if you

are working with non-human species, as other species have different

nucleosome spacing. These values will be updated with EM training, unless

the –f option is set to false. Also note that the first value, corresponding to

the short read distribution, is NOT updated with EM training and will remain

as it is set with this option. It is recommended to set this first value to the

read length, if the read length is under 100bp.

-s , --stddev <Double> Comma separated list of initial standard deviations

values for the fragment distributions., used to create the signal tracks. The

default values are 20,20,20,20. These values may also need to be updated

for non-human species, although it may not be necessary, as the EM is robust

in handling variances. These values are also updated with EM training. Note:

the first value is not updated, as the short distribution uses an Exponential

distribution that only uses the mean value, so the first value is meaningless.

-f , --fragem <True || False> Boolean to determine whether fragment EM

training is to occur or not. The default is true. If this option is set to false,

the initial values described with –m and –s are used as the parameters of

the mixture model. This is generally not recommended. Setting this value

to false can decrease the total runtime, but may result in a less accurate

model. If the data has been run already, and it is desired to re-run it using

different reporting thresholds, you could set this option to false, provided

that you reset the initial parameters to the updated ones created by the

previous run. These values are recorded in the .log file described later.

-q , --minmapq <int> This is the minimum mapping quality score for reads

to be used in creating the signal tracks. The default is 30. Although this can

be set to higher or lower values, it is generally not recommended. If few

meet this threshold, that would indicate that the assay itself was

compromised, and setting a lower value would likely still result in errors.

-u , --upper <int> Upper limit on fold change range for choosing training

regions. HMMR chooses training regions by finding genomic loci whose

fold change above genomic background is within a certain range. This

option sets the upper limit of this range. Default is 10. Generally speaking,

the higher this range is, the more stringent the resulting model, and the

lower the range, the less stringent the model. If recall (sensitivity) is

important for you, it is recommended to set lower ranges, while if

precision or accuracy is more important, it is recommended to set higher

ranges. Once the regions within a certain range are found, HMMR extends

the regions by +- 5KB and uses these extended regions to train the model.

The search ends once a maximum of 1000 regions are identified.

-l , --lower <int> Lower limit on fold change range for choosing training

regions. HMMR chooses training regions by finding genomic loci whose

fold change above genomic background is within a certain range. This

option sets the lower limit of this range. Default is 2. Generally speaking,

the higher this range is, the more stringent the resulting model, and the

lower the range, the less stringent the model. If recall (sensitivity) is

important for you, it is recommended to set lower ranges, while if

precision or accuracy is more important, it is recommended to set higher

ranges. Once the regions within a certain range are found, HMMR extends

regions by +- 5KB and uses these extended regions to train the model.

The search ends once a maximum of 1000 regions are identified.

-z , --zscore <int> Zscored read depth to mask during Viterbi decoding.

Default is 100. In our experience, Viterbi has trouble decoding regions that

Have very high read coverage. If encountered, Viterbi will either call a

Empty block, where no state is called, or will call one large open region.

To avoid this problem, HMMR will skip over any regions whose centered

Read coverage is equal to or greater than this value. If the report peaks

Option is set (-p True – described below), these regions are added back to

The peak file and are labeled as “High\_Coverage\_Peak\_#”. These regions

Can therefore be examined further, if desired.

-o , --output <Name> The prefix for all output files. Default is “NA”.

-e , --blacklist <BED> BED file of regions to exclude from model creation

and genome decoding. These could be previously annotated blacklist

regions. It is always recommended to include a blacklisted region list.

-p , --peaks <True || False> Boolean to determine whether or not to report

a peak file in BED format. Default is True. The resulting file will be in

gappedPeak format and will be described in detail below.

-k , --kmeans <int> Number of states in the model. Default is 4. It is

generally not recommended to change this setting. All of our tests and

comparisons were done using the default setting. It may be reasonable

to set to a lower number of states when using 500 cell per-replicate data,

but this is untested. If this setting is changed, it is generally recommended

to not report peaks, but only to report the genome-wide state annotation

file. This can then be interpreted manually.

-t , --training <BED> BED file of training regions to use instead of using

fold-change ranges. If this option is used, it is recommended to only use

1000 regions and to extend them by +- 5KB, as HMMR would do for

fold-change identified training regions.

--bedgraph <True || False> Boolean to determine whether a genome-wide

state annotation bedgraph file should be reported. Default is False.

--minlen <int> Minimum length of an open region state to call a peak.

Note: that the –p option must be set to true. Default is 200.

--score <max || ave || med || fc || zscore || all> What type of score system

to use for scoring peaks. “Max” refers to the maximum number of

reads mapping to the open region. “Med” refers to the median number

of reads mapping to the open region. “Ave” refers to the average

number of reads mapping to the open region. “FC” refers to fold-change

and is the average number of reads mapping to the open region divided

by the genome average. “Zscore” refers to the average number of reads

mapping to the open region minus the genome average divided by the

genomic standard deviation. “All” reports all these scores separated by

a “\_” (underscore). Default is “max”.

--bgscore <True || False> Boolean to determine whether to add a score to

every state annotation in a bedgraph file. Note that –bedgraph has to be set

true. This adds considerable time to the program and is generally not

recommended. Default is false.

--trim <int> How many signals, or distributions, to trim from the signal

tracks. It trims from the end of the matrix (IE 1 means trim the tri signal

track, 2 means trim the tri and di signal tracks…etc). This could be useful

if your data doesn’t contain many longer fragments. It is always

recommended to try to run at default before attempting this. Default is

0.

--window <int> Size of the bins to split the genome into for Viterbi decoding.

To save memory, HMMR splits the genome into these sized bins and

Performs Viterbi decoding on these bins separately. Default is 25000000.

It may be necessary to reduce the size of these bins, if running HMMR on

A desktop or a machine with limited memory. Most desktops should handle

A bin size of 1/20 the default. A java heap space runtime error, is common

When the bin size is too large for the machine.

--model <File> This model (binary model generated by previous HMMR run,

suffixed with .model – see section 3.3) will be used to decode the genome rather than building a new model using training regions (either provided by user with the –t option or created by HMMR using the –u and –l options). This can be used in conjunction with the –modelonly option (create the model, inspect it and run HMMR with that model).

--modelonly <True || False> Boolean to determine if HMMR should quit after

generating the model. This is a helpful option when trying to determine the

best parameters for creating the model. Default = false.

-h , --help This flag prints a help message and exits the program.

**Section 3: Overview of Output Files**

1) **Name.log** This is a log file generated as HMMR runs. It contains all inputted arguments, the results of the fragment distribution EM algorithm, and various status updates as HMMR progresses, as well as a textual description of the generated model. When finished, it also reports the total amount of actual (real-world) time it took for HMMR to run.

2) **Name.bedgraph** This is the genome-wide state annotation file created if the –-bedgraph option is set to true. It is a four column, tab-delimited file, with the following fields:

Column 1: Chromosome Name

Column 2: Region Start (zero-based)

Column 3: Region Stop (zero-based)

Column 4: State annotation. This represents what state the particular region is assigned to, corresponding to the created model. The state name is prefixed with the letter “E”, to make extractions easier. For instance, to retrieve all regions that were classified as state 1, type

**$ grep E1 Name.bedgraph > State1\_regions.bed**

3**) Name.model** This file contains the model that was generated by HMMR and was used to decode the genome. It is always recommended to check this file after running HMMR. Please note that this file is a binary file that can only be read by HMMR. To see the actual textual description of the model, see the log file (see section 3.1). If HMMR runs to completion but produces poor results, it is usually the result of a faulty model. For instance, if the model shows “NaN” for all of the parameters, such as transition or emission probabilities, this usually means that something went wrong with either the choice of training regions or generating the signal tracks. This can occur with too low or too high of a fold-change range for choosing training regions, if the BED file of training regions was faulty or if certain signals need to be trimmed off (due to small numbers of larger fragments). Generally speaking, this poor model occurs when there is not enough separation in the signal tracks between the states. Common fixes include, trimming the signal tracks (option –-trim), changing the number of states (option –k), choosing a different fold change range for training site identification (option –u and –l) or changing the list of previously annotated training regions (option –t). Additionally, checking the model ensures that the predicted model is created. Although rare, it is possible for a different state to be better indicative of the open state, than is normally the case (HMMR sorts the model so the last or highest numbered state should be the open state). If this happens, then the resulting peak file may be incorrect. In such cases, it is recommended to report a bedgraph file and extract the proper state (of certain lengths and with the accompanying flanking regions) and manually create a peak BED file. We have tested numerous ATAC-seq datasets, using our default parameters and we generally don’t encounter problems, unless we radically change those settings.

1. **Name\_peaks.gappedPeak** This is the standard peak file reported by HMMR by default (when the –p , --peaks option is set to true). The first line in the file is a track line for genome browsers. After that, it is a 15 column, tab-delimited file containg the following fields:

Column 1: Chromosome Name

Column 2: Peak Start (zero-based). This is the beginning of the regulatory region that HMMR identifies as a peak. This includes the flanking nucleosome regions. Therefore, this position represents the start of the upstream nucleosome. If there is no upstream nucleosome, this position represents the start of the open state.

Column 3 Peak Stop (zero-based). This is the end of the regulatory region that HMMR identifies as a peak. This includes the flanking nucleosome regions. Therefore, this position represents the end of the downstream nucleosome. If there is no downstream nucleosome, this position represents the end of the open state.

Column 4: Peak Name. Unique name for the peak, in the format “Peak\_#”. For the high coverage regions that are excluded from Viterbi decoding (described in Section 2, under the –z , --zscore option), the name is in the format “HighCoveragePeak\_#”. This allows the high coverage peaks to be easily identified and extracted.

Column 5: Unused, denoted with “.”

Column 6: Unused, denoted with “.”

Column 7: Open state start (zero-based). This is the genomic position where the open state begins. It is therefore possible to use only the open regions, rather than the regulatory regions that HMMR reports by default.

Column 8: Open state stop (zero-based). This is the genomic position where the open state ends. It is therefore possible to use only the open regions, rather than the regulatory regions that HMMR reports by default.

Column 9: Color code for display. Set to 255,0,0. This would create a dark-blue color on a genome browser.

Column 10: The number of sub-regions in the peak region. For standard peaks this is set to 3, indicating the open state region and the two flanking nucleosome state regions. For high coverage peaks, this is set to 1. Peaks that don’t have upstream and/or downstream nucleosomes will have different values as well.

Column 11: Comma separated list of sub-region lengths. The first and last values are always set to one, making visualization easier. The middle value is the length of the open state region. For high coverage peaks, this is only a single value, denoting the length of the total region. If the peak lacks upstream and/or downstream nucleosomes, these values will reflect that.

Column 12: Comma separated list of the (zero-based) starts of each sub-region, relative to the total region’s start. For instance, the first value is always 0, meaning 0 bp from the region start (column 2). The second value is the distance from the region start (column 2) to the open region start (column 7), etc.

Column 13: Peak Score. This is the score for the peak, as determined by the scoring system option described in Section 2 (option –-score).

Column 14: Unused, denoted as -1.

Column 15: Unused, denomted as -1.

1. **Name\_summits.bed** This is a BED file containing the summits of each HMMR peak. Summits are determined by finding the position within the open state region, whose Gaussian-smoothed read coverage is the maximum over the entire region. This is only outputted if a peak file is also outputted. If motif analysis is being conducted, it is recommended to use these summits as the points of interest, as the summits tend to be better indicative of TF binding as opposed to peak centers. This is a four column, tab-delimited BED file containing the following fields:

Column 1: Chromosome Name

Column 2: Summit Start (zero-based)

Column 3: Summit Stop (zero-based)

Column 4: Summit Name. This is the same name as the corresponding Peak name used in the peak file (column 4 of the Name\_peaks.gappedPeak file).

Column 5: Peak Score. Same score as in the .gappedPeak file. (See section 3.4)