Documentation for EMSAR

version 2.0.1

Soohyun Lee[1,#,+], Chae Hwa Seo[2,#], Burak Han Alver[1], Sanghyuk Lee[2,3,*], Peter J. Park[1,4*],

[1]Center for Biomedical Informatics, Harvard Medical School, Boston, MA, USA
[2]Emerging Technology Center, DNA link, Seoul, South Korea
[3]Ewha Womans University, Seoul, Korea
[4]Informatics Program, Boston Children's Hospital and Division of Genetics, Brigham and Women's Hospital, Boston, MA, USA

[#] These authors contributed equally.
[*] Corresponding authors.
+duplexa@gmail.com

Contents

Overview 1
Downloading & Installation 2
Building an rsh index (emsar-build) 2
  Getting a transcriptome reference 3
  Multi-threading option 3
Main program (emsar) 3
  Multi-threading option 4
  Alignment 4
  Note on alignment with mismatches 5
  Transcriptome fasta file 5
  Streaming of alignments 5
  Multi-sample option 5
  Library type 6
  Maximum number of alignments per read 6
  Segment information 6
  Output logging 7
utils 7
  FPKM2gFPKM.pl 7
  transcript_stats.3.pl 8
  post_processing.pl 9

Overview

EMSAR is a C program that quantifies gene/transcript expression from RNA-seq data. It first builds an rsh (Read-SHaring) index from a transcriptome fasta file given a specified library type and read length (eg. Paired-end, unstranded, 101 bp), then uses it for rapid processing of alignments to estimate the expression levels. The rsh index identifies 'segments' of sequence as either unique or shared among different genes and mRNA isoforms. The expression level of a given transcript can then be estimated from the observed counts of reads supporting its constituent segments, and the length of these segments as computed during indexing.

EMSAR consists of two parts: emsar-build and emsar. The first builds an rsh index, while the second takes alignment files and produces expression estimates. Alternatively, the user can also build an rsh index internally by starting with a transcriptome fasta file when running emsar on RNA-seq read alignments. In this case, read length is automatically obtained from the first bam file and the user can choose either to create an rsh index file or to use the information only internally. It is highly recommended to use emsar-build to create a reusable rsh index file for each read length being used, as this greatly saves time at later steps. The choice of when to do the rsh indexing will not affect the actual quantification result (unless the user specifies incorrect read length or library type when building the rsh index).

If you use EMSAR in your paper, please cite Lee et al. (http://www.biomedcentral.com/1471-2105/16/278)

Downloading & Installation

EMSAR can be downloaded from https://github.com/parklab/emsar/releases. Installation can be done by uncompressing the tar.gz file and typing make.

```
cd emsar-2.0.1
./configure
make
sudo make install
```

Or, if you want to specify a path for it to be installed (e.g. because you don't have permission to the root), do the following (replace 'path' with the real directory name (e.g. /home/username/bin/emsar-2.0.1/).

```
cd emsar-2.0.1
./configure --prefix=path
make
make install
```

Then, if you want, include your emsar-2.0.1 directory to your PATH environmental variable.

Alternatively, download binary files (emsar and emsar-build).

Building an rsh index (emsar-build)

To build an rsh file, the user must specify a transcriptome fasta file. Note that this fasta file must be identical to the one used for mapping RNA-seq reads. Otherwise, the final estimation accuracy may be lower. EMSAR will not compare the two transcriptomes for identity, so it is the user's responsibility to ensure this.

The read length must also be specified. For a single-end library, a range of read lengths can be set. Note that the run time increases proportional to the range of read lengths specified, though specifying a wider range of read lengths would allow the index to be used more broadly. For a single-end library, fragment length may not be specified.

For a paired-end library, only a single read length may be used; it must be identical for all RNA-seq data and it should match the length specified for the rsh index. For paired-end libraries, run time is proportional to the range of fragment lengths (optionally specified by the user). By default, the range of paired-end fragments is 1-400 bp.

Since the run time is sensitive to the chosen range of fragment length, users wishing to specify a new range should consider the true range in their data. RNA-seq fragments beyond the specified range will not be included for later calculation. Since index construction is optimized for multi-threading, users are recommended to take advantage of this performance increase, especially for paired-end experiments.

Additionally, the user must specify an output directory and an output file prefix.

The following shows an example run.

```
emsar-build -P myTranscriptome.fa 101 myOutdir myOutprefix
```

This command will create an rsh file, `myOutdir/myOutprefix.rsh`, for paired-end (`-P`), unstranded (default) 101 bp x 2 reads, covering fragment lengths of 1-400 bp (default). As an example, `myOutprefix` could be `myTranscriptome.L101x2`.

Getting a transcriptome reference

One can directly download a transcriptome reference provided by ENSEMBL (http://useast.ensembl.org/info/data/ftp/index.html) or UCSC Genome Browser (http://genome.ucsc.edu/cgi-bin/hgTables). Another way is to generate it from a genome reference and a gtf file using the gffread program that comes as a part of the Cufflinks package (http://cole-trapnell-lab.github.io/cufflinks/). Alternatively, any other custom-built transcriptome sequence can be used (e.g. a diploid transcriptome that includes both maternal and paternal sequences, a de novo transcriptome built by an assembly program, a transcriptome with additional transcripts such as fusion transcripts, etc.).

It is recommended that the transcriptome reference does not contain polyA sequences. Although it is possible for EMSAR to use a polyA-attached reference, the speed will be greatly reduced, and it is not clear that using such a reference would result in better accuracy.

Multi-threading option

It is recommended that the user take advantage of the multi-threading option for optimal computation speed. Multi-threading can be turned on with a `-P` option. An example run using 8 threads would look like:

```
emsar-build -P -B -p 8 myTranscriptome.fa 101 myOutdir myOutprefix
```

Main program (emsar)

EMSAR requires an rsh index file and an alignment file (bam, sam, or bowtie1 default output format) as input. It assumes that the alignment and rsh indexing are performed on the same transcriptome fasta file (not on the genome). If two different files are used, the program may produce an unpredictable outcome. Likewise, the read length and library type (paired-end vs single-end, unstranded vs forward-stranded vs reverse-stranded) must match. Alternatively, the transcriptome fasta file can be fed directly to the main program (`-x`) instead of an rsh index file (`-I`), but this method will take much longer because the index will be reconstructed for each sample.

Additionally, the user must specify an output directory and an output file prefix.

A simple example run would look as follows:

```
emsar -P -B -I myTranscriptome.rsh myOutdir myOutprefix myPESample.bam
emsar -P -B -x myTranscriptome.fa myOutdir myOutprefix myPESample.bam
```

Here, `myOutdir` and `myOutprefix` must represent the corresponding samples.

Note that `-P` option is used for a paired-end sample and `-B` option is used for a bam file. By default, the library is assumed to be single-end, unstranded, and the default alignment format is bowtie1 output format. For a sam file with a single-end library, EMSAR could be run as follows:

```
emsar -S -I myTranscriptome.rsh myOutdir myOutprefix mySESample.SAM
```

Multi-threading option

As with the index-building program, it is recommended that the user take advantage of the multi-threading option for faster performance. Multi-threading can be turned on with a `-p` option. An example run using 8 threads would look like:

```
emsar -P -B -p 8 -I myTranscriptome.L75x2.rsh myOutdir myOutprefix
myPESample.bam
```

Alignment

Theoretically, any transcriptome-aligned alignment file can be used, but for a good performance, it is recommended to use the following parameters with bowtie1.

```
-a -v 2 -X 1000 -m 100
```

The `-a` and `-v 2` options are recommended, since EMSAR internally filters for best alignments assuming that all alignments (up to a certain number of mismatches) are reported. Although bowtie does a similar filtration with the options of `-a -v 2 --best --strata`, we noticed that the program misses some alignments for paired-end data; in theory, these bowtie options may still be used to allow slightly higher speed for EMSAR though it is less recommended. For this filtration, EMSAR assumes an auxiliary field in the bam or sam file (the MD field) that reports the number of mismatches. Bowtie1 default output format contains this information, which is again used by EMSAR. It is recommended to use ungapped alignment, since EMSAR is not designed to handle insertions and deletions.

The `-X` option specifies the maximum fragment length in bp for paired-end data. It is usually safest to set this number to a sufficiently large number (e.g.1000) even if the actual fragment size is smaller than that. Note that EMSAR will only use fragments within the range used to build the rsh index. If the fragment lengths tend to be larger, then one can use the `-F` (`--maxfraglen`) option of either emsar-build or emsar to increase the maximum fragment size. For single-end data, fragment length-related parameters can be ignored.

The `-m 100` option suppresses alignments if more than 100 alignments exist for a read (or read pair). Similarly, the `-k` option of emsar-build and emsar will internally filter out alignments above a threshold number (default 100). If using the `-m` option to run bowtie, the safest choice is a value equal to or larger than the `-k` value set by emsar-build and emsar.

Note on alignment with mismatches

In rare cases, it is possible that a read may map ambiguously to two locations based on mismatches at different positions in the read. These two locations cannot have identical sequences, therefore they will not appear in the same segment. These and other rare cases, in which alignments allowing mismatches are in conflict with the segments defined using perfect-matches, are excluded from the calculation.

Transcriptome fasta file

We have noticed that a polyA-attached transcriptome fasta files makes the suffix array sorting very slow, and the results based on polyA-attached transcripts are not well-tested. Therefore, it is not recommended to use a polyA-attached fasta file. EMSAR can handle a fasta file containing duplicated sequences, but in that case, the user must be cautious when interpreting the final FPKM, TPM (transcripts per million), or read count values.

Streaming of alignments

An alignment file can be piped directly from an alignment program. For example, if the user begins with fastq files and wishes to align them to a transcriptome index using bowtie1, the command would look like:

```
bowtie -a -v 2 -m 100 myTranscriptomeIndex mySample.fastq  | emsar -I
myTranscriptome.rsh myOutdir myOutprefix
```

Note that the transcriptome index file must have been built from the same transcriptome fasta file. The above command assumes a single-end library. To stream input as a sam or bam file, one could use the following command:

SAM :

```
bowtie -a -v 2 -m 100 -S myTranscriptomeIndex mySample.fastq  | emsar
-S -I myTranscriptome.rsh myOutdir myOutprefix
```

BAM :

```
bowtie -a -v 2 -m 100 -S myTranscriptomeIndex mySample.fastq  |
samtools view -hbS - | emsar -B -I myTranscriptome.rsh myOutdir
myOutprefix
```

Multi-sample option

A user can specify a text file containing a list of alignment files instead of a single alignment file, along with the `-M` option (`--multisample`). For example, the following command would run EMSAR in multi-sample mode.

```
emsar -P -B  -M -I myTranscriptome.rsh myOutdir myOutprefix
myBam.list.txt
```

`myBam.list.txt` could look like:

```
myInputdir/myPESample1.bam
myInputdir/myPESample2.bam
myInputdir2/myPESample3.bam
```

Note that the `-P` and `-B` options are used to match the type of alignment files provided in the list file. Currently, a multi-sample run with mixed library types or alignment file formats is not supported. Alignment streaming is not supported in the multi-sample mode.

Library type

The strandedness of the library can be specified with the `-s` (`--strand_type`) option. By default, '`ns`' (unstranded) is used for both single-end or paired-end libraries. For a stranded single-end library, one can use '`ssf`' (forward stranded) or '`ssr`' (reverse stranded). The former assumes that the reads are mapped to the sense strands of the transcriptome. The latter assumes antisense. For a paired-end library, '`ssfr`' or '`ssrf`' can be used. '`ssfr`' assumes that the first mate is forward and the second mate is reverse; '`ssrf`' is the opposite. An example usage is provided below:

```
emsar -P -B -s ssfr -I myTranscriptome.rsh myOutdir myOutprefix
myStrandedPESample.bam
```

Maximum number of alignments per read

The `-k` (`--max_repeat`) option of EMSAR sets the maximum number of alignments per read. Any read with more than this number will be ignored for both sample read counts and for computing segment lengths. If one uses bowtie for alignment, the `-m` option can be used to restrict the number of alignments per read. If this is set to be smaller than the `-k` option of EMSAR, the accuracy may be slightly sacrificed because the maximum number of alignments for computing virtual length is different from that of the sample. Therefore, it is recommended that the user either matches the `-m` option of bowtie with `-k` option of EMSAR, or make sure that the `-m` option is set to be larger than the `-k` option. A similar recommendation would also apply for other aligners.

For example, if the user wants to include reads mapped to up to 3000 locations, then the bowtie and EMSAR commands could look like:

```
bowtie -S -a -v 2 -m 3000 -X 1000 myTranscriptomeIndex -1
mySample.R1.fastq -2 mySample.R2.fastq | samtools view -hbS - >
mySample.bam
emsar-build -P -k 3000 myTranscriptome.fa myOutdir myOutprefix
emsar -P -B -k 3000 -I myTranscriptome.rsh myOutdir myOutprefix
mySample.bam
```

Segment information

The list of all segments with their associated sequence-sharing set and transcripts (separated by '+') is generated with the `-g` option. The resulting '.segments' file also contains the effective length of each segment for each transcript (separated by comma), and the observed read count and expected read count for each segment. The expected read count is based on the estimated FPKM values and the effective lengths. An example '.segments'

file looks like :

```
segment_id  sequence_sharing_set_id      transcript_ids  transcript_names       eff.length    Readcount
expected_Readcount
c0      s0      t0      ENSVPAT00000007931      474.005893      39      39.000000
c1      s1      t1      ENSVPAT00000004546      0.000000        0       0.000000
c2      s2      t2      ENSVPAT00000003367      1720.000000     84      84.000000
c3      s3      t3      ENSVPAT00000009296      608.011785      59      59.000000
….
c12916  s2644   t4871,t7233,t8590       ENSVPAT00000004400+ENSVPAT00000000132+ENSVPAT00000010274       20.000000  0
4.250379
c12917  s2644   t4871,t8238,t8590       ENSVPAT00000004400+ENSVPAT00000004654+ENSVPAT00000010274       1.639434      1
0.306191
c12918  s307    t4892,t7897,t11970      ENSVPAT00000004120+ENSVPAT00000008179+ENSVPAT00000001600       5.005893      0
0.950490
```

A segment with a single transcript (eg. the first line, segment_id c0) represents the region unique to that transcript. The effective length of this segment (eff.length) is the length of the region unique to this transcript; this quantity is measured as the number of possible alignments unique to this transcript weighted by the fragment length distribution. Likewise, a segment associated with multiple transcripts (eg. the last line, segment_id c12918) represents the region shared exclusively by the listed transcripts and the eff.length is the effective length of this shared region. Readcount and expected_Readcount represent the observed and expected read counts for the designated region, respectively.

An example command to generate this segment information file would look like:

```
emsar -P -B -g -I myTranscriptome.rsh myOutdir myOutprefix
mySample.bam
```

Output logging

An updating progress bar is printed to standard output when the `-v` (verbose) option is used. It is highly recommended not to use this option when standard output is logged. By default, major steps are logged with time stamps. Logging can be turned off using the `-q` option.

utils

The following utils are provided as a part of the EMSAR package.

FPKM2gFPKM.pl
transcript_stats.3.pl
post_processing.pl

FPKM2gFPKM.pl

usage : FPKM2gFPKM.pl g2tfile FPKMfile

This script sums the isoform-level estimates to obtain gene-level estimates of expression. Any gene-to-isoform

mapping can be used. For example, one can use a tab-delimited text file containing an ENSEMBL gene ID and an ENSEMBL transcript ID per line (.g2t file) or a file containing a gene symbol and an ENSEMBL transcript ID per line (.s2t file). The transcript IDs are the same as the ones used in the fasta file and the gene IDs (symbols) represent the target for computing combined expression levels. The example files are shown below.

An example ENSEMBL .g2t file:

```
ENSG00000213366 ENST00000369829
ENSG00000224246 ENST00000448586
ENSG00000034677 ENST00000522182
ENSG00000160323 ENST00000371910
ENSG00000136883 ENST00000491059
ENSG00000054690 ENST00000559981
ENSG00000258905 ENST00000553993
```

An example ENSEMBL .s2t file:

```
GSTM2   ENST00000369829
SFTA2   ENST00000448586
RNF19A  ENST00000522182
ADAMTS13        ENST00000371910
KIF12   ENST00000491059
```

An example usage would look like:

```
FPKM2gFPKM.pl myTranscript.g2t mySample.fpkm > mySample.gfpkm
```

The output file of the script would look as below:

```
geneID  FPKM    iReadcount      iReadcount.int TPM
RP11-560A15.3   0       0       0       0
RPS11   1168.165255     19756   19756   2114.136652
CREB3L1 90.215036       9361.145276     9361    163.270489
RPL10P14        0       0       0       0
```

transcript_stats.3.pl

usage : transcript_stats.3.pl fastafile segmentfile g2tfile

The script generates a series of traits associated with each transcript, including GC content, number of isoforms, and statistics to report how this transcript overlaps with other isoforms or other genes. These overlaps are reported as the effective length unique to a transcript ('isoform_unique_length'), the effective length unique to a gene ('gene_unique_length'), and the effective length shared with other genes ('multi_gene_length'), as well as their proportions relative to the total effective length of this transcript ('total_effective_length'). In order to use this script, one should first run EMSAR with the `-g` option, to create a .segments file which is used as one of the input files here. The other two input files are the transcriptome fasta file and the g2t file (described above). An example output file is shown below (with line numbers to help tracking lines):

```
1: transcript_id   gene    transcript_length       GC_content      nIsoforms       total_effective_length
isoform_unique_length   gene_unique_length      multi_gene_length
```

```
gene_unique_isoform_common_length      isoform_unique_proportion      gene_unique_proportion
gene_unique_isoform_common_proportion
2: ENST00000369829 GSTM2   914    0.509      15     753.258783     322.000000     752.626423
0.63236 0      0.427      0.999      0
3: ENST00000598454 LAMTOR5-AS1    711    0.495       13     551.023467     373.157403     549.943391
1.080076       0      0.677      0.998      0
4: ENST00000448586 SFTA2   451    0.556      12     0      0      0      0      0      NA     NA
NA
5: ENST00000522182 RNF19A  544    0.522      15     384.023466     165.269720     384.023466     0
0      0.430      1      0
6: ENST00000371910 ADAMTS13       1074    0.583      17     914.023467     151.000000     914.023467
0      0      0.165      1      0
```

An example usage would look like:

```
transcript_stats.3.pl myTranscriptome.fa mySample.segments
myTranscript.g2t > mySample.stats
```

post_processing.pl

usage : post_processing.pl fpkm_file_dir g2t_file

The script converts all the isoform-level fpkm files (direct output of EMSAR) in a specified directory into gene-level fpkm (gfpkm) files. Then, given a g2t file (described above), the script merges the gene-level read counts and TPM values of multiple samples into two joint tables (gReadcount.all and gTPM.all).