What is ESTmapper?

ESTmapper is a software package for the high-throughput alignment of large cDNA (EST, mRNA) sequence sets to a large eukaryotic genome of the same species. It can also be used to map relatively short DNA segments, such as sequencing reads or SNP flanking sequences, to the genome.

How does ESTmapper work?

For each cDNA in the input set, ESTmapper detects one or several occurrences of the cDNA in the target genome. ESTmapper uses a three-stage process to locate a cDNA sequence in the genome:

Stage 0
- **Configuration** examine the genomic sequences and pre-compute search indices. This needs to be done once per genome.
Stage 1
**Signal finding** uses an efficient sequence similarity search to identify regions on the genome potentially containing the cDNA (signals).

Stage 2
**Signal filtering** discards regions containing weak signals based on the extent of the cDNA matched and the number of candidate regions.

Stage 3
**Signal polishing** generates a spliced alignment between the cDNA and each of the genomic regions selected earlier.

**Basic usage**

**Configuration**

The first step is to pre-process the genomic sequences, using the `configureESTmapper.pl` program.

There are two required parameters:

- `-genome g.fasta` the genome to map to
- `-genomedir genome-directory` the directory to save the configuration in

And numerous optional parameters:

- `-mersize m` use m-mers (default 20)
- `-merskip s` skip s m-mers between mers (default 0, use all mers)
- `-memory M` use M MB memory for the search processes (default 1000MB)
- `-segment S` use S search processed (default, decided on memory size)
- `-sge` compute the configuration on the grid; args are passed to qsub
- `-sgename` sge job name (default 'EMconfig')
- `-local` compute the configuration right now (the default)

This will build an index into the genomic sequences `g.fasta` allowing fast access to each sequence. It will segment the sequences into equal sized pieces, either sized so that the search processes are expected to run in $M$MB of memory, or into $S$ pieces. A rule of thumb is that each base will need 10 bytes of memory; mammalian genomes comfortably fit in $S=32$ pieces.

The mersize $m$ and merskip $s$ roughly control the space/time/sensitivity of the search process. Smaller mersizes require more time to find signals, generate more false positives (more space to store results, much more time to polish). Larger merskips quickly reduce memory requirements ($s=1$ decreases the number of mers by 50%) which can reduce sensitivity, but doesn't impact search time much.

**Configuration Example**
Configure the *Apis mellifera* genome for ESTmapper, using the default \( m=20 \) mersize and \( s=0 \) merskip. Request four segments, instead of limiting each segment to a specific memory size. Perform the configuration computations using SGE, passing some SGE-specific options to the submission command.

```
configureESTmapper.pl \ 
  -genome /genomedir/ame_ref.fasta \ 
  -genomedir /scratch/apis \ 
  -mersize 20 \ 
  -merskip 0 \ 
  -segments 4 \ 
  -sge "-pe thread 2 -A estmapper"
```

**Basic usage (mapping)**

Once a genome is configured, any number of ESTmapper mappings can be performed, on any input cDNA, EST or short genomic region input set.

**Required Parameters**

- `outputdir`
  - mapping-directory, the output and temporary files will go here.
- `genomedir`
  - genome-directory, as configured with `configureESTmapper.pl`.
- `mapX`
  - mapping style; -mapest, -mapmrna or -mapsnp.

There are numerous optional parameters, grouped by ESTmapper phase:

**Global Parameters**

- `runlater`
  - prepare the computations (search, filter, polish) but do not actually compute them. Instead, tell the user what to do.

**SGE Parameters**

- `sge N`
  - run ESTmapper using SGE, with name N.
- `sgeoptions "O"`
  - "O" is supplied to all SGE submit commands.
- `sgesearch "O"`
  - "O" is supplied to search phase SGE submit commands.
- `sgefilter "O"`
  - "O" is supplied to filter phase SGE submit commands.
- `sgepolish "O"`
  - "O" is supplied to polish phase SGE submit commands.
- `sgefinish "O"`
  - "O" is supplied to finish phase SGE submit commands.
A note about SGE options: It is very important to quote the options to these parameters, otherwise, ESTmapper will attempt to interpret them. For example: -sgepolish "-q idle.q -p -10".

**Search Stage Parameters**

- `localsearches N` compute the search phase locally, on this machine, running $N$ processes at the same time.
- `searchthreads T` use $T$ compute threads per search process.
- `mermaskfile M.fasta` read $M$.fasta, and ignore any mer present there when searching for signal.
- `merignore n` ignore any mers that occur $n$ times or more in the genome.

**Filter Stage Parameters**

- `hitsortmemory M` use $M$ MB memory to sort hits.
- `nofilter` do no filtering of hits.

**Polishing Stage Parameters**

- `relink R` Sim4 'relink' constant, to allow for large introns. The default values are 500 for ESTs and 1000 for mRNAs.
- `alwaysprint A` always report A alignments per query, regardless of quality.
- `batchsize B` run polishing in batches of $B$ signals each.
- `numbatches N` run polishing using $N$ batches of signals.
- `localpolishes N` run polishing locally, on this machine, running $N$ processes at the same time.
- `interspecies` attempt to align between closely-related species.
- `[no]aligns` [do not] save alignments in the output.
- `abort` abort alignments that look suspicions and are expensive.
- `yn` save the answer for each signal processed (for debugging).

Usually, the [sim4] flavor of the -min* options are unnecessary. By default, sim4 will look for alignments 5% below what is specified, e.g., "-mincoverage 50" implies "-sim4mincoverage 45".
If any -min option is supplied, it is better to supply all three. Supplying just "-minlength 100" will leave the default values of "-mincoverage 50" and "-minidentity 95"; especially -mincoverage is probably not desired in this case.

**Termination Stage Parameters**

- `[no]cleanup`  
  [do not] attempt to cleanup the results, removing small spurious exons, among other things
- `savetemporary`  
  save all the gory intermediate files

**Example**

Using the configured *Apis mellifera* genome from the last section, loosely map human mRNA.

```
ESTmapper.pl \
  -outputdir apismap \
  -genomedir /scratch/apis \
  -mapmrna /home/work/SEQUENCE/HUMREFSEQ1/HUMREFSEQ1.fasta \
  -minidentity 80 \
  -mincoverage 50 \
  -minlength 0 \
  -verbose \
  -stats \
  -sge emApis \
  -sgeoptions "-pe thread 2 -A estmapper" \
  -nocleanup \
  -savetemporary
```

This found alignments for (the number of alignments is the first number):

<table>
<thead>
<tr>
<th></th>
<th>gi</th>
<th>ref</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11038618</td>
<td>NH_001614.2</td>
<td>actin, gamma 1 (ACTG1), mRNA</td>
</tr>
<tr>
<td>19</td>
<td>24371439</td>
<td>NR_000037.1</td>
<td>tRNA glutamine 1 (TRQ1) on chr 17</td>
</tr>
<tr>
<td>5</td>
<td>32526874</td>
<td>NR_001449.1</td>
<td>tRNA lysine 1 (TRK1) on chr 17</td>
</tr>
<tr>
<td>3</td>
<td>5016088</td>
<td>NM_011011.2</td>
<td>actin, beta (ACTB), mRNA</td>
</tr>
<tr>
<td>4</td>
<td>58293775</td>
<td>NR_002213.1</td>
<td>Transfer RNA arginine (TRR) on chr 3</td>
</tr>
<tr>
<td>3</td>
<td>62990120</td>
<td>NM_00107421.1</td>
<td>actin-like protein (FKSG30), mRNA</td>
</tr>
<tr>
<td>1</td>
<td>68299777</td>
<td>HM_001069.2</td>
<td>tubulin, beta 2A (TUBB2A), mRNA</td>
</tr>
<tr>
<td>9</td>
<td>73760420</td>
<td>NR_002457.1</td>
<td>tRNA proline 2 (TRP2) on chr 14</td>
</tr>
</tbody>
</table>

**Input/Output**

The **input** consists of two large multi-fasta files, containing the cDNA and the genomic sequences, respectively.

The **output** consists primarily of the final alignment and summary files, located in the work directory:

- polishes-good  
  all cDNA-genomic alignments that meet the specified quality criteria
- polishes-best  
  for each EST, only the 'best' genomic alignment among those that meet the specified quality criteria  
  (if multiple alignments with the same 'best' characteristics exist, then all are reported)
- summary  
  summary mapping statistics for the run
The following associated sequence files will also be created:

cDNA-good.fasta
  multi-fasta file of input sequences that were successfully mapped
cDNA-missing.fasta
  multi-fasta file of input sequences that had signals, but did not produce a valid match
cDNA-zero.fasta
  multi-fasta file of input sequences that did not have any signals

**ESTmapper output format**

sim4begin
edef=cDNA define
ddef=genomic define
cDNAbgn_1-cDNAend_1 (GENbgn_1-GENend_1) <M-N-P> intronOrientation
cDNAbgn_2-cDNAend_2 (GENbgn_2-GENend_2) <M-N-P> intronOrientation
...
cDNAbgn_n-cDNAend_n (GENbgn_n-GENend_n) <M-N-P> intronOrientation
cDNA alignment sequence for exon #1
genomic alignment sequence for exon #1
cDNA alignment sequence for exon #2
genomic alignment sequence for exon #2
...
cDNA alignment sequence for exon #n
genomic alignment sequence for exon #n
sim4end

where:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cDNAidx</td>
<td>internal index of the cDNA in the input cDNA fasta file</td>
</tr>
<tr>
<td>cDNAlen</td>
<td>length of the cDNA sequence</td>
</tr>
<tr>
<td>pA(T)</td>
<td>length of polyA(T) tail detected and masked</td>
</tr>
<tr>
<td>GENidx</td>
<td>internal index of the genomic sequence in the genome fasta file</td>
</tr>
<tr>
<td>GENoff</td>
<td>offset to the beginning of the genomic region containing the signal</td>
</tr>
<tr>
<td>GENlen</td>
<td>length of the genomic region containing the signal</td>
</tr>
<tr>
<td>M</td>
<td>number of nucleotide matches in the alignment</td>
</tr>
<tr>
<td>N</td>
<td>number of matching N's in the alignment</td>
</tr>
<tr>
<td>P</td>
<td>percent sequence identity of the alignment</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$O$</td>
<td>match orientation:</td>
</tr>
<tr>
<td></td>
<td>- <strong>forward</strong> the cDNA sequence aligns to the genomic sequence directly</td>
</tr>
<tr>
<td></td>
<td>- <strong>complement</strong> the reverse complement of the cDNA sequence matches the genomic sequence; this is the equivalent of the Sim4 'complement' output line</td>
</tr>
<tr>
<td>$S$</td>
<td>strand predicted based on the splice signals and alignment quality:</td>
</tr>
<tr>
<td></td>
<td>- <strong>forward</strong> high alignment quality, predicted forward strand</td>
</tr>
<tr>
<td></td>
<td>- <strong>reverse</strong> high alignment quality, predicted reverse strand</td>
</tr>
<tr>
<td></td>
<td>- <strong>unknown</strong> low alignment quality or weak splice signals</td>
</tr>
<tr>
<td>$cDNA_{bgn_i}$</td>
<td>start position of exon $i$ in the cDNA sequence</td>
</tr>
<tr>
<td>$cDNA_{end_i}$</td>
<td>end position of exon $i$ in the cDNA sequence</td>
</tr>
<tr>
<td>$GEN_{bgn_i}$</td>
<td>start position of exon $i$ in the genomic sequence (interval $GEN_{lo}$-$GEN_{hi}$)</td>
</tr>
<tr>
<td>$GEN_{end_i}$</td>
<td>end position of exon $i$ in the genomic sequence (interval $GEN_{lo}$-$GEN_{hi}$)</td>
</tr>
<tr>
<td>$M$</td>
<td>number of nucleotide matches in the alignment</td>
</tr>
<tr>
<td>$N$</td>
<td>number of matching N's in the alignment</td>
</tr>
<tr>
<td>$P$</td>
<td>percent sequence identity of the alignment</td>
</tr>
<tr>
<td>$\text{intronOrientation}$</td>
<td>predicted orientation of the intron:</td>
</tr>
<tr>
<td></td>
<td>- $\rightarrow$ forward (i.e., GT-AG-like splice signals)</td>
</tr>
<tr>
<td></td>
<td>- $\leftarrow$ reverse (i.e., CT-AC-like splice signals)</td>
</tr>
<tr>
<td></td>
<td>- $\sim$ ambiguous</td>
</tr>
<tr>
<td></td>
<td>- $\equiv$ gap (unaligned portion) in the cDNA sequence</td>
</tr>
</tbody>
</table>

Exon coordinates are nucleotide based, starting from 1. Genomic coordinates are always in the original sequence, while the cDNA coordinates will refer to positions in the reverse complement of the sequence if the match orientation is indicated as 'complement'.

Lowercase letters in the alignment lines indicate positions with matching nucleotides, ‘-’ indicate gaps in the corresponding sequence, and uppercase letters mark either substitutions, or gaps in the other sequence.

**Output Format Examples**

A minimal match description. The cDNA and genomic sequence deflines and alignment lines need not be present:

```plaintext
sim4begin
2[2472-0-0] 0[239821074-51250] <2472-0-100-complement-reverse>
1-1542 (238923075-239824616) <1542-0-100> <-
1543-1794 (239827671-239827922) <251-0-99> <-
1795-1954 (239834345-239834504) <160-0-100> <-
1955-2472 (239869807-239870324) <518-0-100>
sim4end
```
The record describes a cDNA-genomic alignment between the cDNA with index number 2 (the third sequence in the cDNA multi-fasta file) and the genomic segment between positions 239821075-23987233. The header line contains 3 tokens that provide information about the cDNA (2472 bp long, no polyA or polyT tails identified), the genomic sequence (index 0, and range), and the match (number of nucleotide identity matches -- 2472, percent sequence identity -- 100%, match orientation -- complement, and likely gene location, on the reverse strand).

The alignment contains 4 exons. Since the cDNA and the genomic sequences match in 'complement', the cDNA coordinates are given in the reverse complement of the cDNA sequence (e.g., exon 1, 1-1542, refers to the segment between positions 2472-1542+1 and 2472-1+1 in the original sequence).

A more detailed match description, including alignments and deflines.

This is the same match as in example 1. The edef line now characterizes the cDNA sequence (i.e., the RefSeq sequence corresponding to the opsin 3 gene mRNA), and the ddef line characterizes the genomic sequence (i.e., human chromosome 1). The alignment contains one substitution (C-A) at position 3 in exon 2.

Intermediate Data Files

ESTmapper creates several intermediate data files located in the designated working directory as follows:

0-input
symbolic links to input files, indices and other temporary files
1-search
temporary directory for Stage I (signal finding)
2-filter
temporary directory for Stage II (signal filtering)
3-polish
temporary directory for Stage III (alignment and validation)

Software and hardware requirements

ESTmapper is written in a combination of Perl and C++, and should run on any Posix compliant unix-like platform. For large tasks, such as mapping 6M EST sequences to the human genome, at least 1GB RAM and up to 80GB disk for intermediate results is required; the output will be approximately 12 GB (9 GB
for 'polishes-good', 3 GB for 'polishes-best'.

ESTmapper was designed, developed and implemented by Brian Walenz (brianwalenz@sourceforge.net) and Liliana Florea (floreald@sourceforge.net).

Retrieved from "http://two/wiki/index.php/ESTmapper_Documentation"

- This page was last modified 06:03, 26 October 2007.
- Content is available under Attribution-Noncommercial-No Derivative Works 3.0 Unported.