Transcriptome database roadmap

Christian Iseli, LICR-ITO

0.5, January 14, 2003

Abstract

None yet...

1 Rough introduction about the data

The 3'tags were extracted from about 2'400'000 trace files, by extracting the 50 nt that lay directly before the longest polyA tract in the trace. A minimal length of 10 As was enforced.

The genomic (DNA) data used are the NCBI human chromosome contigs `NT_*`. Since the contigs can be quite large, they are split in chunks of 1.1 megabases, with 100 kilobases of overlap. Thus for example, contig `NT_011515.6`, which is 3427677 nt long, will be split into 4 pieces:

- `NT_011515_0` will be `NT_011515[1..1100000]`;
- `NT_011515_1` will be `NT_011515[1000001..2100000]`;
- `NT_011515_2` will be `NT_011515[2000001..3100000]`;
- `NT_011515_3` will be `NT_011515[3000001..3427677]`.

Note that a contig less than 1100001 nt long will not be split.

Transcribed RNA data are extracted from several sources:

- human EST section of EMBL
- human HTC section of EMBL
- human mRNA documented in the human section of EMBL
- ORESTES sequences from the LICR/FAPESP Human Cancer Genome project
- human mRNA documented in the NCBI curated RefSeq database
- published CHR21 gene list
- SEREX sequences

The majority of ORESTES sequences are also found in the EMBL EST databanks, but it appears a few tens of thousand did not get included yet. So, I use the EMBL entries when available, and resort to our private ORESTES collection for the others. All ORESTES entries are marked differently in the mapping output to indicate that they are not derived from oligo-dT primed cDNAs and might be more prone to DNA contamination.

The transcript sequences are filtered against vector, E. coli, and mitochondrial sequences. The repetitive and RNA elements are masked out. The filtering and masking is performed using the PFP software package (Paracel, Pasadena, CA).

`megablast` (from the NCBI blast package[2]) is used to identify pair-wise similarities between all transcript sequences and the genomic data. The parameters used for megablast are `-f T -J F -F F -W 48`, so we expect near perfect matches on at least 48 nt.

For each pair of matching RNA and genomic sequences, local alignments are generated using a locally modified version of sim4[1], with parameters `W=15 R=0 A=4 P=1 I=12`. The output of `sim4` is filtered as explained in section 2.

The 3'tags are mapped to the genomic data, allowing for 1 or 2 mismatches on the 50 nt length of the tag.

CpG islands are located with a CpG island-specific profile and the program pfsearch from the pftools package, implementing the multiple match search method for generalized profiles[3].

The in-built graphical displays of Acedb[4] are used to create an integrated view of the data. Simple parsing scripts are used to convert the filtered `sim4` output, 3'tag information, RepeatMasker (A. F. Smit, Institute for Systems Biology, unpublished) output, CpG island predictions, GenScan[5] predictions, and genomic sequence data into ace format, suitable for uploading to Acedb. Arrangement of the data within Acedb is controlled by assigning a method (determining position and display color) to each data set, with minor modifications to the sequence class model. The original data is visible within Acedb via the LongText functionality.

2 Filtering the `sim4` output

We use a locally modified version of `sim4`. The modifications were performed to better control the splicing according to `GT/AG`, `GC/AG`, and `AT/AC` splicing rules, and provide more informative output. The script `Fdosim4` is used to call and parse the output of the `sim4` program.

The filtering is done on two different levels: the first level decides if one or more exons should be removed from the map, and the second level decides if the whole map should be rejected altogether. The filtering mechanism is conditioned by the expected quality of the RNA sequence used to produce a given map. The RNA elements are split into two categories:

trusted RNA elements originating from the HUM and HTC sections of EMBL, as well as those coming from RefSeq, which are accepted as reliable.

non-trusted EST-type elements (ESTs from EMBL, ORESTES, SEREX), which are expected to be of lower quality.

Moreover, EST-type RNA elements are expected to have a varying quality over their entire length. Many EST entries are annotated with the start and stop position of their high-quality part, and `Fdosim4` can load EST quality definition files (option `-eq`). Such a file looks like so:

```
AA471110 1 154
AA471338 1 410
AA542808 1 1
AA542811 1 97
```

The first field is the AC, the second is the start position of the high-quality sequence, and the third is the stop position. These fields are derived from the annotation of the EMBL entries. All EST-type RNA elements are then assigned a high quality sequence start and stop position, as obtained from the file. Those not found in the file receive the default values 1 for the start position and n (option `-H`, default 300) for the stop.

2.1 Filtering maps from trusted RNA elements

The individual exon filtering step ensures that the head and tail exons:

- have a quality of at least 98%;
- do not define introns longer than 50,000 nt;
- define splice junction with good quality (> 8);
- the 3' exon is not abnormally short (< 30 nt).

Below is an example of a map where the first exon was filtered out because:

- its quality is only 65%;
- it defines an intron of 623,369 nt;
- the quality of the splice junction is only 6;
- the 3' exon is only 20 nt long.

```
>chr|NT_011512_12|NT_011512.4 Chromosome 21; [Homo sapiens] ; LEN=1100000
>rna|AL110183|AL110183.1|- [Homo sapiens]Homo sapiens mRNA; DKFZp566A221; LEN=568

X 51906-51919  (1-20)   65% <- (GT/AG) 6
675289-675466  (21-198)   100% <- (GT/AG) 10
676015-676139  (199-323)   100% <- (GT/AG) 10
680420-680590  (324-494)   100% <- (GT/AG) 10
685642-685715  (495-568)   98%

     0     .    :    .    :    .    :    .    :    .    :
 51906 GTTTTTTTTTTTA      CAA...CACTCAACATCACCAAATAATTTA
       |||||||||||| ------<<<...<<<|||||||||||||||||||||
     1 GTTTTTTTTTTTTTTTTTT         TCAACATCACCAAATAATTTA

    50     .    :    .    :    .    :    .    :    .    :
675310 TTTGGACTCAGAATTAAAAGAACATTTGACAGTTATGAAATGCATGTTTA
       ||||||||||||||||||||||||||||||||||||||||||||||||||
    42 TTTGGACTCAGAATTAAAAGAACATTTGACAGTTATGAAATGCATGTTTA

   100     .    :    .    :    .    :    .    :    .    :
```

```
 675360 TTCTGAAACTTCTAACTAGTTGTACAACTAATCCGTGACAAATTACCAGA
        ||||||||||||||||||||||||||||||||||||||||||||||||||
     92 TTCTGAAACTTCTAACTAGTTGTACAACTAATCCGTGACAAATTACCAGA

    150     .    :    .    :    .    :    .    :    .    :
 675410 TTAATTTTACTTTATTTCTTCAGGCCTGGGGTTTTTCGATGACTTCAAAT
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    142 TTAATTTTACTTTATTTCTTCAGGCCTGGGGTTTTTCGATGACTTCAAAT

    200     .    :    .    :    .    :    .    :    .    :
 675460 TTGGGATCTA...TACCTTCAAATTTGAAGGTGGGAAATGTATTCATGTC
        |||||||<<<...<<<|||||||||||||||||||||||||||||||||
    192 TTGGGAT         CTTCAAATTTGAAGGTGGGAAATGTATTCATGTC

    250     .    :    .    :    .    :    .    :    .    :
 676049 TGCATTACCAAACATTTGCTTGAGCTTAAAAAGCTCCCTCTCCAGCTCTT
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    233 TGCATTACCAAACATTTGCTTGAGCTTAAAAAGCTCCCTCTCCAGCTCTT
...
```

The global map filtering ensures that the whole map:

- has an overall quality of at least 98%;
- has no holes;
- covers the whole length of the RNA element, except for the first and last 120 nt (empirical value) to account for possible included polyA tract and/or vector.

No coverage requirement is applied when the RNA element maps near the end (<100,000 nt) of the DNA contig.

2.2 Filtering maps from non-trusted RNA elements

The first filtering step finds the best group of contiguous exons in the map. Then, a set of criteria is used to determine if the remaining exons are “well-behaved”. A well-behaved exon must:

- have a quality of at least 88%;
- not define introns longer than 50,000 nt;
- define splice junction with good quality (> 8);
- not be abnormally short (< 30 nt).

When two, or more, of the above points are not met, the exon is rejected.

Below is an example of a map where the first three exons were filtered out.

```
>chr|NT_011512_0|NT_011512.4 Chromosome 21; [Homo sapiens] ; LEN=1100000
>est|BG284935|BG284935.1|- [Homo sapiens]602409188F1 IMAGE:4538349 5’; LEN=681

X 37202-37222  (1-25)   64% <- (AT/AC) 4
X 37509-37509  (26-26)   100% <- (AT/AC) 6
X 37847-37858  (27-38)   91% -- 4
135348-135986  (39-681)   95%

      0     .    :    .    :    .    :    .    :    .    :
  37202 CCT GCTTTTTTTTTCCCCAGG   GTC...TTTCTTT...TATTCTTTT
        |||-|  ||||||||||||   ---<<<...<<<|<<<...<<<| ||||
      1 CCTCGGGTTTTTTTTCCCCCCCCCC         C         TTTTTT

     50     .    :    .    :    .    :    .    :    .    :
  37853 TTTTTTTTT...AACCCTCTTTATATTTGTAATGTTTGTCTCCAGAGTAC
        ||||||===...===  | |||||||||||||||||||||||||| |||
     33 TTTTTT         TTTTTTTATATTTGTAATGTTTGTCTCCAGAATAC

    100     .    :    .    :    .    :    .    :    .    :
 135383 TC   TT  TTTTAAGGGTTTAAATTTTCCAAGGTCTTTCAAAAGTAATT
        ||---||--|||||||||||||| ||||||||||||||| || |||||||
     74 TCTTTTTTATTTTAAGGGTTTATATTTTCCAAGGTCTTTGAACAGTAATT
```

```
    150     .    :    .    :    .    :    .    :    .    :
 135428 ACATTTATAATAATTTTATTAAGTATGAACTTCCTGATGTTGACTATGAT
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    124 ACATTTATAATAATTTTATTAAGTATGAACTTCCTGATGTTGACTATGAT
...
```

The global map filtering ensures that the whole map has an average quality of 93% over its high-quality part, and 88% over the remainder.

3 About `tromer`

`tromer` is an attempt to automate the reconstruction of transcripts from the mapping of RNA data to the DNA contigs. Each map output by `sim4` is used to construct a graph. The graphs originating from each single `sim4` map are then merged when either they share a common edge, or one exon from a graph is contained in the exon of another graph.

Once all possible merging has been done, `tromer` outputs the collection of merged graphs, along with predicted transcripts.

3.1 Synopsis

This is the usage information printed by the `tromer` program. Most of the options are described in more details in the following sections.

```
Usage: /home/chris/src/isrec/cluster/tromer [options] -t <tagDB> <file>...
where options are:
 -L <int>   maximal intron length at ends of a DNA contig [100000]
 -r <int>   merge map stops and splice within range [10]
 -J <int>   for EST with a known 3'tag: adjust the map if the
            mapping end is less than this distance apart from
            the 3'tag site [50]
 -M <path>  write selected maps in specified file [none]
 -o <path>  produce output in specified file(s)
            a %s is replaced by the chromosome name [none]
 -g <path>  produce graph information in specified file(s)
            a %s is replaced by the chromosome name [none]
 -c <path>  clone insert length is read from specified file.
            Can be supplied several times.
 -C <int>   default clone insert length [2000]
 -a <int>   re-assemble contig pieces according to FASTA header
            where the pieces are separated by the specified
            number of nucleotides [0].  A 10% overlap between
            pieces is assumed
 -p <name>  listen on this port [don't listen]
```

3.2 Operation

Upon startup, `tromer` reads in map files derived from `sim4` output and produced by the `Fdosim4` script. Here is an example map:

```
>chr|NT_030187_0|NT_030187.1 Chromosome 21; [Homo sapiens] ; LEN=229586
>est|AA969880|AA969880.1|- [Homo sapiens]op41h11.s1 IMAGE:1579461 3'; LEN=460
92944-93150  (7-212)   98% -> (GT/AG) 10
101963-102108  (213-358)   100% -> (GT/AG) 10
110878-110976  (359-460)   97%

      0     .    :    .    :    .    :    .    :    .    :
  92944 CATACACTCCGTCTCCTGAAGGGGAAGCGGGCTCTTCTCAGATGCACAGG
        |||||-   ||||||||||||||||||||||||||||||||||||||||
      7 CATAC GAACGTCTCCTGAAGGGGAAGCGGGCTCTTCTCAGATGCACAGG

     50     .    :    .    :    .    :    .    :    .    :
  92994 GACAATGTGAAAATCCTGTCCTCAGATTGAGAGGCTGTTTCGTGGGCACC
```

```
        ||||||||||||||||||||||||||||||||||||||||||||||||||
     56 GACAATGTGAAAATCCTGTCCTCAGATTGAGAGGCTGTTTCGTGGGCACC

    100     .    :    .    :    .    :    .    :    .    :
  93044 GAATTCGGGGTCAGGAAAGCAGCCTGCATCCACGAGTATCCTCGGGTTAC
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    106 GAATTCGGGGTCAGGAAAGCAGCCTGCATCCACGAGTATCCTCGGGTTAC

    150     .    :    .    :    .    :    .    :    .    :
  93094 TAAGTGGGGCCAGTGGCTCCAGGTGTAACCCATTTAAGTTTGCCAGACAG
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    156 TAAGTGGGGCCAGTGGCTCCAGGTGTAACCCATTTAAGTTTGCCAGACAG

    200     .    :    .    :    .    :    .    :    .    :
  93144 CCGGGATGTA...AAGGTTCTGGTGAAGGATCTGAAGTGTATGGCCACAC
        |||||||>>>...>>>|||||||||||||||||||||||||||||||||
    206 CCGGGAT         GTTCTGGTGAAGGATCTGAAGTGTATGGCCACAC

    250     .    :    .    :    .    :    .    :    .    :
 101997 CAGTCCCAGAAGAGCCTGGGAGAAGGGAAGATGGTGAACACAGTGGAGTT
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    247 CAGTCCCAGAAGAGCCTGGGAGAAGGGAAGATGGTGAACACAGTGGAGTT

    300     .    :    .    :    .    :    .    :    .    :
 102047 CTGCTGCAAAGCCGAAGATGGTTCTGGCACGTGGCATGACCCACATGACT
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    297 CTGCTGCAAAGCCGAAGATGGTTCTGGCACGTGGCATGACCCACATGACT

    350     .    :    .    :    .    :    .    :    .    :
 102097 CAACATCAGGAGGTA...CAGATCTGACTTCATAAAAGTGAACTATCACA
        ||||||||||||>>>...>>>|||||||||||||||||||||||||||||
    347 CAACATCAGGAG         ATCTGACTTCATAAAAGTGAACTATCACA

    400     .    :    .    :    .    :    .    :    .    :
 110907 ATGCTGCTTTGCAAGCTGTGTGTGAGTGTAAAAGCGTTGAAACTTCCTCA
        ||||||||||||||||||||||||||||||||||||||||||||||||||
    388 ATGCTGCTTTGCAAGCTGTGTGTGAGTGTAAAAGCGTTGAAACTTCCTCA

    450     .    :    .    :
 110957 ATAAATGAAAAGATATCTTT
        ||||||||||||||||||---
    438 ATAAATGAAAAGATATCTTTAAA
```

`tromer` parses this input and keeps only part of the information. It creates a structure to describe each

- DNA sequence: AC, length, chromosome;
- RNA sequence: AC, length, type (EST, HTC, ORESTES, mRNA, RefSeq, Celera, SEREX, other), read direction (3' or 5'), clone name.

For each map, it creates a structure that describes which DNA matches with which RNA, on which strand, and the start and stop points of all the exons. It also records the exon quality, and the direction, type, and quality of the splice junctions.

If `tromer` is asked to assemble the contig pieces (option `-a`), it will combine the maps from the RNA elements that map on several pieces of the same DNA contig.

`tromer` then loads the 3'tags database, parsing entries of the form:

```
ID   3P039433
SQ    1  0 CAAATACTACTGTAGACCCCAGTGTTTATTCATTAAATTTTTTAAATATTtgtttta...
QQ   A:3  C:0  G:3  T:9  test:ok
CM   xy20b05.x1:AW474790-  15 As in polyA
GN     AC073135_1 - pos:  58255 CHR: NA
GN     AC069287_2 + pos:   2714 CHR: 21
GN     AL391217_9 + pos:   1899 CHR:  1
SQ    2  6 CTACTGTAGACCCCAGTGTTTATTCATTAAATTTTTTAAATATTTGTTTTatttgga...
```

```
QQ   A:4  C:2  G:3  T:6  test:ok
CM   qq01c11.x1:AI337492-  26 As in polyA
GN     AC073135_1 - pos:  58249 CHR: NA
GN     AC069287_2 + pos:   2720 CHR: 21
GN     AL391217_9 + pos:   1905 CHR:  1
SQ    3 12 TAGACCCCAGTGTTTATTCATTAAATTTTTTAAATATTTGTTTTATTTGGaatccat...
QQ   A:4  C:2  G:2  T:7  test:ok
CM   op32d11.s1:AA978124-  18 As in polyA
CO   yd74d08.s1:T79778- zf41c03.s1:AA705748-
GN     AC073135_1 - pos:  58243 CHR: NA
GN     AC069287_2 + pos:   2726 CHR: 21
GN     AL391217_9 + pos:   1911 CHR:  1
//
```

A structure is created for each entry. A tag map structure is created for each trusted (marked `test:ok`) match on a DNA element. Each tag map entry describes which DNA matches with which tag, on which strand, and the minimum and maximum positions of the tag cluster.

The `sim4` maps loaded previously involving ESTs corresponding to the trusted tags are updated with the defined 3'-most position in the 3'tags.

3.3 Map filtering

`tromer` filters all the loaded `sim4` maps. For each RNA element, the idea is to decide to which contig it really maps.

The filtering is done in three steps. The first step gets rid of all maps where the RNA element is not of experimental origin (e.g., Celera). The second step purges the remaining maps from trusted RNA elements (mRNA, RefSeq, etc.) where the mapping is only partial. These occur when the map was kept at the `Fdosim4` stage because it was located near the end of a contig piece, and now that the contig has been reassembled it is apparent that the coverage is only partial.

The last step assigns an overall quality to each map by summing up the weighted length of the defined exons, the weight being given by the `sim4` percent quality, and divide it by the covered length. The maps with a score within 1% of the maximum score are kept, and the others discarded.

3.4 Graph generation

Each map is converted into a graph. For example, given the map below:

```
>chr|NT_030187_0|NT_030187.1 Chromosome 21; [Homo sapiens] ; LEN=229586
>est|AA969880|AA969880.1|- [Homo sapiens]op41h11.s1 IMAGE:1579461 3'; LEN=460
92944-93150  (7-212)   98% -> (GT/AG) 10
101963-102108  (213-358)   100% -> (GT/AG) 10
110878-110976  (359-460)   97%
...
```

`tromer` will produce the following graph:

```
H0 [92944]
 =E0=> D0
D0 [93150]
 H0 =E0=>
 -I0-> A0
A0 [101963]
 D0 -I0->
 =E1=> D1
D1 [102108]
 A0 =E1=>
 -I1-> A1
A1 [110878]
 D1 -I1->
 =E2=> T0
T0 [110976]
 A1 =E2=>
```

This graph is composed of six nodes (H_0, D_0, A_0, D_1, A_1, and T_0) and of five edges (E_0, I_0, E_1, I_1, and E_2). There are four possible node types:

H (for head) denote the start of a transcript (but not necessarily the transcription start site);

D are splice donor sites;

A are splice acceptor sites;

T (for tail) denote the end of a transcript (but not necessarily the transcription stop site).

There are two possible edge types:

E are exons;

I are introns.

The graph description above shows that node H_0 has no incoming edge and one outgoing edge, which is exon E_0 going to node D_0; node D_0 has one incoming edge: exon E_0 coming from node H_0, and one outgoing edge: intron I_0 going to node A_0. The rest should be self-evident...

3.5 EST graph merging through clone information

When several EST elements originate from the same clone, according to their annotation, `tromer` attempts to merge their graphs into a consolidated graph.

For example, given the two maps below:

```
>chr|NT_030187_0|NT_030187.1 Chromosome 21; [Homo sapiens] ; LEN=229586
>est|W37332|W37332.1|- [Homo sapiens]zc11h06.r1 IMAGE:322043 5'; LEN=584

153748-154331  (1-584)   98%


>chr|NT_030187_0|NT_030187.1 Chromosome 21; [Homo sapiens] ; LEN=229586
>est|W37838|W37838.1|+ [Homo sapiens]zc11h06.s1 IMAGE:322043 3'; LEN=400

153024-153372  (1-353)   98%
```

and given that the EST annotation pretends that the clone insert length is 1440 nucleotides long, `tromer` deduces that the EST clone named 322043 is fully collinear with the DNA from `NT_030187.1` from position 153024 to position 154331, and creates a new graph to record that information. The clone insert length information is passed to `tromer` using the `-c` switch. When no information is available, `tromer` assumes a default length of 2000 nucleotides (`-C` switch).

3.6 Graph merging

The graphs originating from each single `sim4` map are merged in two steps. The first step merges two graphs when:

- they share a common edge;
- one exon from a graph is contained in the exon of another graph.

The second step tries to extend the 3' end of graphs by merging them with overlapping graphs from unspliced RNA elements.

Finally, graphs consisting of a single exon which did not originate from a trusted (non-EST) RNA element are discarded.

3.7 Graph output

In the graph output file (specified using the `-g` option), `tromer` will output a detailed description of graphs remaining after the merge steps. Here is an example graph:

```
>map|NT_030188_43|NT_030188.1|- Chromosome 21 515779..520531
H0 [520531] 0,1
 =E0=> D0 6,-145
D0 [520311] 1,1
 H0 6,-145 =E0=>
 -I0-> A0 6,-60
H1 [519480] 0,1
 =E1=> D1 3,-63
A0 [519462] 1,1
 D0 6,-60 -I0->
 =E2=> D1 8,-187
D1 [519313] 2,1
```

```
 H1 3,-63 =E1=>
 A0 8,-187 =E2=>
 -I1-> A1 10,-100
H2 [517075] 0,1
 =E3=> D2 16,-163
A1 [517064] 1,1
 D1 10,-100 -I1->
 =E4=> D2 11,-252
D2 [516918] 2,1
 H2 16,-163 =E3=>
 A1 11,-252 =E4=>
 -I2-> A2 25,-249
A2 [515978] 1,1
 D2 25,-249 -I2->
 =E5=> T0 25,-373
T0 [515779] 1,0
 A2 25,-373 =E5=>

E0 520531..520311 0,6
 E:AA307808 1..221 (520531..520311)
 E:AW950156 1..220 (520530..520311)
 E:BI517365 1..103 (520413..520311)
 E:BI517974 540..438 (520413..520311)
 E:BI760118 1..100 (520410..520311)
 R:NM_005423 1..79 (520389..520311)
E1 519480..519313 0,3 I0
 E:AW028846 436..335 (519411..519313)
 E:BG183703 21..158 (519450..519313)
 E:BI764458 1..170 (519480..519313)
E2 519462..519313 0,8
 E:AA307808 222..371 (519462..519313)
...
 R:NM_005423 80..229 (519462..519313)
E3 517075..516918 0,16 I1
 E:AA741431 310..210 (517019..516918)
...
 E:BG222052 7..157 (517065..516918)
E4 517064..516918 0,11
 E:AA307808 372..437 (517064..516999)
...
 R:NM_005423 230..376 (517064..516918)
E5 515978..515779 1,25
 3P058337 515792..515795
 E:AA741431 209..26 (515978..515792)
...
 E:BI764458 318..507 (515978..515789)
I0 520310..519463 6
 E:AA307808 221..222 GT/AG -10
 E:AW950156 220..221 GT/AG -10
 E:BI517365 103..104 GT/AG -10
 E:BI517974 438..437 GT/AG -10
 E:BI760118 100..101 GT/AG -10
 R:NM_005423 79..80 GT/AG -10
I1 519312..517065 10
 E:AA307808 371..372 GT/AG -10
...
 E:BI764458 170..171 GT/AG -10
 R:NM_005423 229..230 GT/AG -10
I2 516917..515979 25
 E:AA741431 210..209 GT/AG -10
...
 E:BI764458 317..318 GT/AG -10
```

The graph output consists of two parts: a description of the graph structure, which has already been partly described in section 3.4, and a detailed list of the exons and introns composing the edges of the graph.

In the graph description, each node is followed by its position on the DNA contig, and by the number of incoming and outgoing edges. Each edge is marked with the number of its contributing RNA elements, and the total weight of their strand indication.

Each exon is then listed, with its start and end position on the DNA contig, followed by the number of 3'tag falling within its bounds, and by the number of associated RNA elements. In the above graph, exon E_5 starts at position 515978 and ends at position 515779 of the DNA contig `NT_030188.1`, contains 1 3'tag (namely `3P058337`) at position 515792, and has 25 associated RNA elements. Each RNA element is listed individually on a line. For example, the first element in exon E_0 from the example above

```
E:AA307808 1..221 (520531..520311)
```

`E:` means it is an EST element, `AA307808` is its accession number, `1..221` the RNA nucleotides used in this exon, and `(520531..520311)` the corresponding nucleotides on the DNA. The defined codes for the RNA elements are as follow:

E EST

O ORESTES

S SEREX

M mRNA from EMBL

R RefSeq

H HTC from EMBL

U other

L EST clone (think link), as explained in section 3.5

Potential alternative splicing, as evidenced by overlapping exon and intron edges, are output as a list of conflicting introns after the exon description. In the example above, exon E_1 is annotated with an overlapping intron (I_0), and E_3 with I_1.

The part of the RNA elements associated with each exon is then explicitly listed.

Each intron is then listed, with its start and end positions on the DNA contig, followed by the number of associated RNA elements. Similarly to the exon listing, the part of the RNA elements associated with each intron is then explicitly listed.

3.8 Transcript generation

Each graph is then used to produce a set of putative transcripts in the output file (specified using the `-o` option). The idea is to produce enough transcripts to cover all the edges of the graph at least once. The generation of a transcript is a three step process: select a seed edge, extend toward the 5' end, and extend toward the 3' end.

The seed edge is first selected among unused 5'-most exons, then among any unused edge.

The extension process always attempts to include unused edges which were derived from the same RNA elements as the seed edge.

Using the example graph of section 3.7, `tromer` produces the following three transcripts:

```
>chr|NT_030188|NT_030188.1 Chromosome 21; LEN=1378645
>map|NT_030188_43_0|43_0|- 515779..520531 E5,I2,E4,I1,E2,I0,E0 3P058337; LEN=718
515779-515978 (1-200)  100% <- (GT/AG) 10
516918-517064 (201-347)  100% <- (GT/AG) 10
519313-519462 (348-497)  100% <- (GT/AG) 10
520311-520531 (498-718)  100%

>chr|NT_030188|NT_030188.1 Chromosome 21; LEN=1378645
>map|NT_030188_43_1|43_1|- 515779..520531 E5,I2,E4,I1,E1 3P058337; LEN=515
515779-515978 (1-200)  100% <- (GT/AG) 10
516918-517064 (201-347)  100% <- (GT/AG) 10
519313-519480 (348-515)  100%

>chr|NT_030188|NT_030188.1 Chromosome 21; LEN=1378645
>map|NT_030188_43_2|43_2|- 515779..520531 E5,I2,E3 3P058337; LEN=358
515779-515978 (1-200)  100% <- (GT/AG) 10
516918-517075 (201-358)  100%
```

Each transcript is annotated with the sequence of exons and introns that compose it, and the 3'tags that occur within one of its exons.

The transcript ID (`NT_030188_43_2`) is composed of the DNA ID (`NT_030188`), graph ID (`_43`), and instance number (`_2`).

4 From graphs to genes

Given the biological fact that some genes are duplicated, and the fact that there still might be some redundancy in the assembled genomic contigs used to construct the transcriptome, some mean is needed to associate genes to their corresponding transcribed locii.

Given the way tromer constructs graphs, the graph(s) representing the same gene will receive different, unrelated identifiers each time the graphs are reconstructed.

To solve these issues, and provide a set of stable identifiers for both known and unknown genes, I have created a new program which purpose is to merge together graphs representing the same gene and to assign stable identifiers across different transcriptome database builds. This program is named `grCanon`, to create canonical graphs.

The `grCanon` program bases its decisions solely on the RNA components of each graph. The RNA components are split into three categories:

1. RefSeq elements, which are given the highest priority;
2. other full length mRNA elements;
3. EST-type elements.

Here follows a description of how the program operates.

4.1 Synopsis

```
grCanon [options] <file>...
where options are:
 -a         do not add new stable graphs.
 -c <int>   pseudocount for computing % EST identity [17].
 -l <path>  read stable graphs from this file [none].
 -m <path>  write graph mapping to the specified file [none].
 -o <path>  write stable graphs to the specified file [none].
 -p <int>   min. % EST identity in merged graphs [60].
 -t <str>   stable IDs tag [HTR].
 -v         be verbose.
```

The `grCanon` program reads a list of currently know graphs from a file specified through the `-l` option. These are called stable graphs, and normally represent merged graphs obtained from a previous version of the transcriptome database. One of the goals of `grCanon` is to identify those known, stable graphs in the new transcriptome database. Each stable graph is given an identifier starting with a tag specified by the `-t` option, followed by a six digit number. The updated set of merged, stable graphs will be written to the file specified through the `-o` option, and the mapping between `tromer` graphs and stable graphs will be written in the file specified through the `-m` option.

4.2 Graph pairs scoring

For each pair of graphs G_a and G_b that share at least one RNA element, a multivalued score is computed as follows:

R_{ab} is the number of common RefSeq elements

RM_{ab1} is the number of RefSeq elements found in G_b but missing in G_a

RM_{ab2} is the number of RefSeq elements found in G_a but missing in G_b

M_{ab} is the number of common full length mRNA elements

MM_{ab1} is the number of mRNA elements found in G_b but missing in G_a

MM_{ab2} is the number of mRNA elements found in G_a but missing in G_b

E_{ab} is the number of common EST-type elements

EM_{ab1} is the number of EST elements found in G_b but missing in G_a

EM_{ab2} is the number of EST elements found in G_a but missing in G_b

Then a pair of similarities for high-quality RNA elements is computed as:

$$HQS_{ab1} = \frac{R_{ab} + M_{ab}}{R_{ab} + M_{ab} + RM_{ab2} + MM_{ab2}}$$

$$HQS_{ab2} = \frac{R_{ab} + M_{ab}}{R_{ab} + M_{ab} + RM_{ab1} + MM_{ab1}}$$

Both HQS_{ab1} and HQS_{ab2} are forced to 0 when $R_{ab} + M_{ab} \equiv 0$.

Similarly, a pair of similarities for EST-type elements is computed as:

$$S_{ab1} = \frac{E_{ab} + C}{E_{ab} + C + EM_{ab2}}$$

$$S_{ab2} = \frac{E_{ab} + C}{E_{ab} + C + EM_{ab1}}$$

where

$$C = \begin{cases} 0 & \text{if } E_{ab} \equiv 0 \\ P & \text{otherwise} \end{cases}$$

where P is a pseudo-count defined by option `-c` which has a default value of 17. This pseudo-count is meant to allow more aggressive merging of graphs with a low number of RNA elements. Both S_{ab1} and S_{ab2} are forced to 0 when $E_{ab} \equiv 0$.

4.3 Graph merging

The merging of two graphs G_a and G_b is allowed or denied based upon the following ordered criterion list:

1. denied if both G_a and G_b are stable graphs (note that when a stable graph is merged with a normal graph, the resulting merged graph is stable);
2. allowed if $R_{ab} > 0$, i.e., they share one RefSeq element;
3. denied if $RM_{ab1} \neq 0$ and $RM_{ab2} \neq 0$, i.e., both have RefSeq elements, but none in common;
4. allowed if $HQS_{ab1} \equiv 100\%$ or $HQS_{ab2} \equiv 100\%$;
5. denied if $M_{ab} \equiv 0$ and $MM_{ab1} \neq 0$ and $MM_{ab1} \neq 0$, i.e., both have full length mRNA but none matches;
6. allowed if $S_{ab1} \geq M\%$ or $S_{ab2} \geq M\%$, where M is the minimum match percent defined by option `-p` which has a default value of 60;
7. denied otherwise.

Graph merging is an iterative process. At each round, the scores for each graph pairs are computed, and the pair with the best score is merged. Here is the method used to compare scores. Suppose we have the scores comparing graphs G_a vs G_b, and G_c vs G_d. We define:

$$X_{ij} = \begin{cases} 1 & \text{if } G_i \text{ and } G_j \text{ can be merged} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{ij} = \begin{cases} \min RM_{ij1}, RM_{ij2} & \text{if } R_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$BR_{ij} = \begin{cases} 1 & \text{if } R_{ij} \equiv 0 \text{ and } RM_{ij1} \neq 0 \text{ and } RM_{ij2} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$QM_{ij} = \begin{cases} 1 & \text{if } \max HQS_{ij1}, HQS_{ij2} \equiv 100\% \\ 0 & \text{otherwise} \end{cases}$$

$$BM_{ij} = \begin{cases} 1 & \text{if } M_{ij} \equiv 0 \text{ and } MM_{ij1} \neq 0 \text{ and } MM_{ij2} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$QE_{ij} = \max S_{ij1}, S_{ij2}$$

and we say that the score of G_a vs G_b is better than the score of G_c vs G_d when one of the following applies, in the given order:

1. $X_{ab} > X_{cd}$
2. $R_{ab} > R_{cd}$
3. $B_{ab} < B_{cd}$
4. $BR_{ab} < BR_{cd}$
5. $QM_{ab} > QM_{cd}$
6. $BM_{ab} < BM_{cd}$
7. $QE_{ab} > QE_{cd}$.

The merging phase completes when no more graphs can be merged. At this point, any remaining unstable graph becomes a stable graph and is given a new stable identifier.

5 What is where

This is an attempt at a roadmap to describe what data is available, where it is stored, and how to access it.

5.1 RNA preprocessing

The RNA data is collected from various local repositories in `/db` and copied on

`ludwig-gm1:/paracel5/scratch/trome/hum`

each in its own subdirectory:

- `est_hum-re/est_hum-re.seq` for the human EST sequences from the EMBL release;
- etc.

The `do_pfp` script, shown in figure 1 applies the Paracel filtering package to each collection, with the parameters specified in the file:

`pfpest_human.prm`

which is given in figure 2. The masked sequences are found in the files:

- `est_hum-re/est_hum-re_m.seq`;
- etc.

```
#!/bin/sh
for f in est_hum-re rna_hum-re rs_hum-re htc_hum-re orestes \
      est_hum-up rna_hum-up rs_hum-up htc_hum-up serex; do
  cd $f;
  rm -rf *.debris stat.out scylla.out scylla.err
  scylla -Query $f.seq -Param ../pfpest_human.prm \
         -Output ${f}_m.seq -Stat stat.out \
         >scylla.out 2>scylla.err </dev/null
  cd ..
done
```

Figure 1: `do_pfp` script

The masked sequence files are used to construct Blast databases on two machines:

- `isrec-insect7` in `/export/scratch`;
- `isrec-insect8` in `/export/scratch/trome`;

The Blast databases are named `est_hum-re`, etc.

5.2 Mapping on the DNA contigs

The source of the genomic DNA data is at the NCBI, in their genomes repository:

`ftp://ftp.ncbi.nih.gov/genomes/H_sapiens`

For each chromosome, I retrieve two files: the assembled contigs in FASTA format (e.g., `hs_chr21.fa.gz` for chromosome 21), and the contig annotations in GenBank format, without the sequence data (e.g., `hs_chr21.gbs.gz` for chromosome 21).

The mapping is then performed on `isrec-insect7` and `isrec-insect8` using the `do_map` script given in figure 3. The script can be split in four parts:

1. uncompress the FASTA file retrieved from the NCBI, and adjust the FASTA headers to my likings. For example, given the NCBI file `hs_chr1.fa.gz` this step will produce the file `chrom1.seq`;
2. split the sequences in 1.1 MB chunks, so that the used tools do not choke to death. This will produce the file of split sequences `chrom1_s.seq` for chromosome 1, etc.;
3. use `megablast` to find high similarity regions to the masked RNA elements. For each type of RNA element, this will produce a file of matches. For example, for chromosome 21, this will produce the following files:

   `chrom21_CRAs` for Celera matches

chrom21_ESTs for EST matches

chrom21_HTCs for High Throughput cDNA matches

chrom21_OREs for ORESTES matches

chrom21_RNAs for matches to mRNA from EMBL

chrom21_RSs for RefSeq matches

chrom21_SEs for SEREX matches;

chrom21_OTRs for other matches;

4. run the `Fdosim4` script to produce maps. For example, for chromosome X this will produce the file `chromX_map`, and the rejected maps will be stored in the file `chromX_map.log`.

5.3 Directory structure of `/db/trome`

Here is a description of the important subdirectories of `/db/trome`.

5.3.1 `CHR_raw`

The `CHR_raw` directory contains the DNA contigs with the fixed headers. There is one FASTA file per chromosome, i.e., chromosome 1 is in the file `chrom1.seq`, etc. The files are indexed by the contig's accession numbers. There is a configuration file for `fetch`, and the way to retrieve, for example, contig `NT_009458` is to use:

```
fetch -c /db/trome/CHR_raw/fetch.conf trome:NT_009458
```

The directory also contains the contig annotations from the NCBI in GenBank format. There is also one file per chromosome: `hs_chr1.gbs.gz` for chromosome 1, etc.

5.3.2 `map`

The `map` directory contains the map files. There are three types of maps, and each type is stored in one file per chromosome:

1. the maps produced by the `sim4` program are stored in files ending in `_map`, i.e., `chrom1_map` for chromosome 1, etc.
2. the transcripts predicted by the `genscan` program are stored in files ending in `_gsc`, i.e., `chrom1_gsc` for chromosome 1, etc.
3. the transcripts constructed by the `tromer` program are stored in files ending in `_tr`, i.e., `chrom1_tr` for chromosome 1, etc.

All the files are in a format similar to the output of the `sim4` program. Each file is accompanied by an index file, ending in `.mix`. Given the following sample map:

```
>chr|NT_011512_0|NT_011512.4 Chromosome 21; [Homo sapiens]
 gi:16170824 Hs21_11669 working draft sequence segment; LEN=1100000
>est|AI672225|AI672225.1|+ [Homo sapiens]wc26b02.x1 NCI_CGAP_Kid11
 Homo sapiens cDNA clone IMAGE:2316267 3', mRNA sequence.; LEN=343

654399-654720  (22-343)   93%

      0     .    :    .    :    .    :    .    :    .    :
 654399 AGAAGTACCAATGGGT GCAGGATTCTTTACCTCAGGACTTTTAAGACCT
        ||||||||-|| ||| -|||||||||||||| ||||||||||||||||||
     22 AGAAGTAC AAGGGGGGGCAGGATTCTTTATCTCAGGACTTTTAAGACCT
...
```

the following accession numbers will be indexed:

- `NT_011512_0`
- `NT_011512`
- `AI672225`

which are the contig's accession number, with and without the split chunk number information, and the accession number of the RNA element. The `fetchmap` script can be used to retrieve maps by using the above accession numbers. Beware that running:

```
fetchmap NT_011512
```

will result in a very large output.

The `map` directory also contains a file named `selected.map`. This file is produced by `tromer`, and defines, for each RNA element, the DNA contig where it belongs. The file is indexed by contig accession number, with and without the split chunk number information. The data can be retrieved using `fetch`, for example:

```
fetch -c /db/trome/map/fetch.conf selected:NT_011512_0
```

5.3.3 `TAG_db`

The `TAG_db` directory contains the 3'tags database, in the file `tags.dat`. The file is indexed by tag accession number, and by DNA contig accession number. To fetch a tag given its accession number, one can use:

```
fetch -c /db/trome/TAG_db/fetch.conf tags:3P000001
```

To fetch all the tags on a given DNA contig, one can use:

```
fetch -c /db/trome/TAG_db/fetch.conf -m tags_gac:NT_011512
```

The `TAG_db` directory also contains the tentative 5'tags database, derived from human EPD entries, in the file `tags_epd.dat`. The file is indexed by tag accession number, and by DNA contig accession number. To fetch a tag given its accession number, one can use:

```
fetch -c /db/trome/TAG_db/fetch.conf tags_epd:5P000001
```

To fetch all the tags on a given DNA contig, one can use:

```
fetch -c /db/trome/TAG_db/fetch.conf -m tags_epd_gac:NT_011512
```

To fetch the tags derived from a given EPD entry, one can use:

```
fetch -c /db/trome/TAG_db/fetch.conf -m tags_epd_eac:EP07111
```

5.3.4 `tromer`

The `tromer` directory contains the graph generated by the `tromer` program, and data derived from the constructed transcripts. The graphs are stored in one file per chromosome, with the graphs of chromosome 1 in the file `chrom1_gr`, etc. Each graph file is indexed by the accession numbers of:

- the graph (e.g., `NT_004377_50`), tag: `graph`
- the ESTs used in the graph, tag: `graph_e`
- the mRNAs used in the graph, tag: `graph_m`
- the RefSeq entries used in the graph, tag: `graph_r`
- the ORESTES used in the graph, tag: `graph_o`
- the SEREX used in the graph, tag: `graph_s`
- the HTCs used in the graph, tag: `graph_h`
- other special RNAs used in the graph, tag: `graph_u`

To fetch a graph, given it accession number, one can use:

```
fetch -c /db/trome/tromer/fetch.conf graph:NT_011512_2
```

To fetch all the graphs using the EST `AI672225`, one can use:

```
fetch -c /db/trome/tromer/fetch.conf -m graph_e:AI672225
```

To see to which graph(s) is assigned your favorite RefSeq sequence, use:

```
fetch -m -c /db/trome/tromer/fetch.conf graph_r:NM_003263
```

And so on, for each index type.

The `tromer` directory also contains predicted transcripts in FASTA format. For each chromosome, there are two files:

`chrom1_r.seq` contains the putative mRNA data;

`chrom1_p.seq` contains putative protein translation of the CDS, produced by `ESTScan` through analyzing the putative mRNA data.

There are also files containing SAGE tag information. For each chromosome, there are two files:

`chrom1_nla` contains data for `NlaIII` SAGE tags;

`chrom1_sau` contains data for `Sau3AI` SAGE tags.

The format of the SAGE tag files is as follows:

```
NT_030187_0_0  453 3P116327 ACTCAACATC  339 ACCCACATGA  330
NT_030187_1_0 2063        - TGCTCCAGTT 1968 AAGGGTGGGG 1899 ATAGCTTCCA 1792
NT_030187_1_1  563        - CAGAACAAAA  405 TCCGGTGGCA  287
NT_030187_2_0  524 3P116329 Gaaaaaaaaa  523 TTGCGGAGAT   92
```

Each line has at most 9 fields, the first 5 are mandatory, and the rest is optional.

1. `tromer` predicted transcript id
2. length of the predicted transcript
3. 3'tag id used as polyadenylation site to detect corresponding SAGE tags. If the transcript does not end at a 3'tag, a dash (`-`) is used
4. first (3' most) SAGE tag
5. position of the first SAGE tag
6. second SAGE tag
7. position of the second SAGE tag
8. third SAGE tag
9. position of the third SAGE tag

In cases where the first SAGE tag of a more upstream 3'tag is the same as the second or third SAGE tag of a downstream 3'tag (or the end of the transcript), the second and/or third SAGE tags are omitted.

In cases where the first SAGE tag of two (or more) 3'tags is the same, the most downstream 3'tag is shown.

When the SAGE tag is near the 3'tag so that part of the tag is As from the polyA tail, the As are shown in lower case (see `NT_030187_2_0` above).

To get a predicted `tromer` transcript in FASTA format, use:

```
fetch -c /db/trome/tromer/fetch.conf rna:NT_006098_115_0
```

To get the `ESTScan` predicted protein:

```
fetch -c /db/trome/tromer/fetch.conf pep:NT_006098_115_0
```

To see the SAGE tags associated with a graph, use:

```
fetch -c /db/trome/tromer/fetch.conf nla:NT_006098_115
```

If you are only interested in a particular transcript, use:

```
fetch -c /db/trome/tromer/fetch.conf nla:NT_006098_115_0
```

You have the choice between `nla` for `NlaIII` SAGE tags, and `sau` for `Sau3AI`.

If you are interested to know which graphs are associated with a given SAGE tag, use:

```
fetch -m -c /db/trome/tromer/fetch.conf nla1:GCCAGGAGGA
```

if you are only interested in the first SAGE tag (3' most), or:

```
fetch -m -c /db/trome/tromer/fetch.conf nlax:GCCAGGAGGA
```

if you want the tag in any of the three positions. Use `nla2` for second and `nla3` for third. Use `sau`, `sau1`, `sau2`, `sau3`, `saux` for `Sau3AI` tags. Once you have the graph, it is easy enough to see which RefSeq and/or mRNA is associated with the given SAGE tag.

All the putative proteins can also be found, in SwissProt format, in the file `trome.dat`.

5.4 Some numbers

Here are some figures based on the NCBI NT contigs of May 2002, using RNA elements from what corresponds fairly closely to EMBL release 71 (i.e., release 70 + all the updates), and the other RNA databases as of May 2002.

The table below shows the number of RNA elements after the different steps of the mapping process:

source total available sequences

PFP after PFP filtering

megablast after `megablast`

sim4 after `sim4`

tromer after `tromer` selection

graph used in one or more graphs

RNA	source	PFP	megablast	sim4	tromer	graph
EST	4371521	4175823	3735913	3624876	3624876	3192497
HTC	3938	3881	3773	3488	3274	3274
orestes	51404	49653	32300	29730	29730	24046
RNA	83087	80721	72263	59516	56307	56307
RS	20051	20047	14975	13861	13342	13342
serex	2496	2424	2287	1882	1882	1784
celera	26544	26527	25979	25152	24983	NA

The `tromer` program generated 110679 graphs.

References

[1] Florea, L., Hartzell, G., Zhang, Z., Rubin, G.M. and Miller, W. A computer program for aligning a cDNA sequence with a genomic DNA sequence, *Genome Res*, 8:967–74, 1998.

[2] Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. A greedy algorithm for aligning DNA sequences, *J Comput Biol*, 7(1-2):203–14, 2000.

[3] Bucher, P., Karplus, K., Moeri, N., and Hofmann, K. A flexible motif search technique based on generalized profiles, *Comput. Chem.*, 20:3–24, 1996.

[4] Kelley, S. Getting started with Acedb, *Briefings in Bioinformatics*, 1:131–137, 2000.

[5] Burge, C., Karlin, S. Prediction of complete gene structures in human genomic DNA, *J Mol Biol*, 268:78–94, 1997.

```
%ATAIL
-Alg           atail
-PolyDist      40
-Threshold     8
-Action        mask
%DUST
-Alg           dust
-Threshold     22
-Action        mask
%VECTOR
-Reference     ../reference/UniVec
-Threshold     30
-Action        mask
-WordLen       8
-Matrix        ../matrix/dna.p1m4.l.mat
%E.coli
-Reference     ../reference/ecoli.fna
-Threshold     40
-Action        filter
-WordLen       12
-Debris        ./human.ecoli.debris
-Matrix        ../matrix/dna.p1m9.l.mat
%MITO
-Reference     ../reference/human.mito
-Threshold     100
-Action        filter
-WordLen       12
-Debris        ./human.mito.debris
-Matrix        ../matrix/dna.p1m9.l.mat
%RNA
-Reference     ../reference/human.RNA
-Threshold     35
-Action        mask
-WordLen       12
-Matrix        ../matrix/dna.p1m9.l.mat
%REPEATS
-Reference     ../reference/human.repeats
-Threshold     187
-Action        mask
-WordLen       8
-Matrix        ../matrix/dna.p9m12TT-30-5.mat
%TRIMJUNK
-Alg           trimjunk
-JunkChars     N
-MaskChar      N
-Threshold     10
-Action        mask
%MINLEN
-Alg           minlen
-JunkChars     NX
-Threshold     50
-Action        filter
-Debris        ./human.short.debris
```

Figure 2: Parts of file `pfpest_human.prm` which defines filtering and masking parameters.

```
#! /bin/sh
if [ $# -ne 2 ]; then
  echo "usage: $0 <chrom nb> <np proc>"
  exit 1
fi
zcat hs_chr$1.fa.gz | \
  sed 's/^>gi|\([^|]*\)|ref|\([^.]*\)\.\([^|]*\)|\([^ ]*\) \
         Homo sapiens chromosome \([^ ]*\)/\
         >chr|\2|\2.\3 Chromosome \5; [Homo sapiens] gi:\1 \4/' \
    > chrom$1.seq
/home/chris/src/isrec/psc/Fsplseq -l 1000000 -o 100000 \
 chrom$1.seq >chrom${1}_s.seq 2>chrom${1}_s.log </dev/null || exit 1
megablast -d "est_hum-re est_hum-up" -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_ESTs </dev/null
megablast -d "htc_hum-re htc_hum-up" -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_HTCs </dev/null
megablast -d "rna_hum-re rna_hum-up" -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_RNAs </dev/null
megablast -d "rs_hum-re rs_hum-up" -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_RSs </dev/null
megablast -d orestes -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_OREs </dev/null
megablast -d serex -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_SEs </dev/null
megablast -d celera -i chrom${1}_s.seq \
 -f T -J F -F F -W 48 -a $2 >chrom${1}_CRAs </dev/null
if [ -s spec${1}.nsq ]; then
  megablast -d spec${1} -i chrom${1}_s.seq \
   -f T -J F -F F -W 48 -a $2 >chrom${1}_OTRs </dev/null
  /home/chris/src/isrec/psc/Fdosim4 -e chrom${1}_ESTs \
   -o chrom${1}_OREs \
   -c chrom${1}_HTCs -x chrom${1}_SEs -v chrom${1}_CRAs \
   -r chrom${1}_RNAs -m chrom${1}_RSs -n chrom${1}_OTRs \
   -b spec${1}.seq \
   -eq /db/trome/stuff/est_hum-re.qual \
   -eq /db/trome/stuff/est_hum-up.qual \
   chrom${1}_s.seq >chrom${1}_map 2>chrom${1}_map.log </dev/null
else
  /home/chris/src/isrec/psc/Fdosim4 -e chrom${1}_ESTs \
   -o chrom${1}_OREs \
   -c chrom${1}_HTCs -x chrom${1}_SEs -v chrom${1}_CRAs \
   -r chrom${1}_RNAs -m chrom${1}_RSs \
   -eq /db/trome/stuff/est_hum-re.qual \
   -eq /db/trome/stuff/est_hum-up.qual \
   chrom${1}_s.seq >chrom${1}_map 2>chrom${1}_map.log </dev/null
fi
```

Figure 3: `do_map` script