trieFinder

Gabriel Renaud

Max Planck Institute for Evolutionary Anthropology

gabriel.reno [at sign] gmail.com

June 17, 2013

Contents

1 Description

trieFinder in summary:

Feature	Description
Input data	MPSS sequence tags (with a common prefix)
Use	align the input, annotate the results and parse them automatically
Language	C++ core functions and Perl wrapper scripts
Input format for sequence tags	fastq or unaligned BAM (see below)
Input format for databases	fasta
Output format	custom tab delimited (see section 4)

trieFinder is a software suite for the automatic alignment of tags stemming from Massively Parallel Signature Sequencing (MPSS) to preannotated databases of putative sequence tags from both the genome and transcriptome (Refseq and Unigene). Thus, the laborious process of aligning the tags and parsing the results to create a final summary for each tag is obviated. The core elements of trieFinder are written in C++ with wrapper scripts in Perl. Although the software was originally written to deal with Zebrafish data, it can be easily accomodated for other species.

Our approach relies on prefix trees built on the database which has the advantage of assuring correct results even if the sequence is shorter than the one stored in the database. Therefore, the user does not have to rebuild the database for each iteration.

1.1 Requirements

trieFinder needs:

- Perl installed
- a C++ compiler
- zlib installed
- The genome and transcriptome (see section 2.2)
- Adequate disk space to store the resulting indices (about 92G for a 76bp zebrafish index)
- Sufficient RAM to store the indices into memory (10-20G should suffice)

1.2 Procedure

trieFinder will:

- Build a database of the putative sequence tags within 1 mismatch of a common prefix (representing the restriction enzyme recognition sequence) for the RefSeq, Unigene and Genomic database
- Combine the putative tags into a single database of all putative sequence tags
- Search the input tags against the single database and produce a final tab delimited file detailing the matches for each sequence tag

To avoid running into the upper limit of the RAM, we divide the last two steps into chunks which are automatically combined later on.

2 Installation

2.1 Building the binaries

Make sure the main script and worker scripts are executable:

```
chmod u+x mainScript.pl
chmod u+x src/mergeAnnotation.pl
chmod u+x src/readFiles.pl
```

Build the C++ binaries:

```
cd src/
cd gzstream/
make
cd ..
make
```

If the following files are present, the build was succesful:

```
src/allTag
src/searchtags
```

2.2 Setting up the databases

Download the databases:

- Download the RefSeq for Zebrafish, for example we used

  `ftp://ftp.ncbi.nih.gov/refseq/D_rerio/mRNA_Prot/zebrafish.rna.fna.gz`

- Download the Unigene for Zebrafish, for example we used

  `ftp://ftp.ncbi.nih.gov/repository/UniGene/Danio_rerio/Dr.seq.all.gz`

- Download the Genome for Zebrafish, for example we used

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

- Concatenate all the genomic sequences for the Zebrafish chromsomes (genomic database) into a single file, this can be done using the unix "cat" command :

  `cat [list of all chromosomes] > whole_genome.fa`

- Unzip any database file you have downloaded as the program requires raw fasta for the database files.

3 Running the software

To run the software type:

`./mainScript.pl`

This will give you a list of options. The first time the software is executed, it needs to build the formatted databases for the search. On subsequent searches, if the same directory where the database files are stored is specified, it will merely use the existing files. Be sure to delete these files if the database you have built is older than 1 year and download the databases from scratch. If you only wish to build the databases, use the "-d" option.
You can use tags against a database that was built for longer tags (e.g. input has 35bp against a database of 76bp) but please note that this may induce false negatives due to tags occuring at the very end of a reference sequence that were too short for inclusion in the database. Although this should not have any significant impact on overall results, it can be easily solved by rebuilding the database for the length of the tags in the input.
The output will be created by adding the ".out" extention to the input fastq file containing the tags that is specified using the "-i" option. If you have an unmapped BAM file, you can use this custom version of samtools (https://github.com/udo-stenzel/samtools-patched) which can produce fastq as output. You can use the output of samtools-patched by specifying the input as such:

```
samtools view -Y input.bam > input.fq
```

If you do not have write permissions in the directory where the input file is stored, you can simply create a symbolic link.

Here is an example of use:

```
./mainScript.pl -r GATC -o /home/user/project/zebrafish/db/
-s /mnt/db/zebrafish.rna.fna -u /mnt/db/Dr.seq.all
-g /mnt/db/whole_genome.fa -i /home/user/project/zebrafish/tags.fq
```

This will build the database for the Sau3Al restriction enzyme (recognition sequence = GATC), will create and store the database files in :

```
/home/user/project/zebrafish/db/
```

Using the following raw fasta files for RefSeq, Unigene and the full zebrafish genome respectively:

```
/mnt/db/zebrafish.rna.fna
/mnt/db/Dr.seq.all
/mnt/db/whole_genome.fa
```

and will read the following fastq file as input:

```
/home/user/project/zebrafish/tags.fq
```

The output will be produced as :

```
/home/user/project/zebrafish/tags.fq.out
```

4 Output description

The output is a tab delimited file where the following columns contain:

Column #	Content
1	Original id of the input sequence
2	Original sequence for this id
3	Perfect matches to the refseq database [1]
4	Number of initial perfect matches to the refseq database
5	Degenerate[2] matches to the refseq database
6	Number of initial degenerate matches to the refseq database
7	Perfect matches to the unigene database
8	Number of initial perfect matches to the unigene database
9	Degenerate matches to the unigene database
10	Number of initial degenerate matches to the unigene database
11	Perfect matches to the genome
12	Number of initial perfect matches to the genome
13	Degenerate matches to the genome
14	Number of initial degenerate matches to the genome

[1] : By default, the program will only report at most 10 matches. The number of original matches will be report in round brackets in the next column.
[2] : Degenerate in this context means matches with 1 or more mismatches.
A match has the following format:

```
id of reference match[:]position (0-based):strand
```

Each match is delimited by three commas (‘,,,‘).