Using the command line tools

Tim Millar

2017-07-31

Contents

TEFingerprint

TEFingerprint is a collection of related tools including:

- `tefingerprint`
- `tef-extract-informative`
- `tef-filter-gff`

The help text can be displayed for any of these tools using the `-h` or `--help` flags e.g.:

```
tefingerprint -h
```

Running the full TEFingerprint pipeline requires the following input data:

- paired-end reads in fastq format (2 files per sample).
- a reference genome in fasta format.
- a reference set of repeat elements in fasta format.
- optionally, an annotation of repeat elements within the reference genome in gff3 format.

The TEFingerprint pipeline requires some consistency with names of repeat elements. It is recommended that you use a naming convention similar to `<super-family>_<family>_<additional-info>` e.g.

Gypsy_26_chr15_18793972 or Gypsy_Gypsy26_chr15_18793972. The chosen naming convention should be used in both the reference set of repeat elements (a fasta file) and the reference annotation of repeat elements (a gff3 file in column 3, the "type" column).

When identifying clusters of selected categories of repeat elements, the start of repeat names from input files is matched to the specified categories of interest. For example if two repeats (transposons) are named Gypsy_2_more-info and Gypsy_7_more-info they will both be identified as being in the Gypsy category. However, they will be placed in separate categories if Gypsy_2 and Gypsy_7 are specified as categories. If a repeat name matches two or more categories it will be placed in the one with the longest name. For example a repeat named Gypsy_27_more-info will be categorised as Gypsy_27 in preference of Gypsy_2.

Extracting informative reads

Identifying informative reads requires a paired-end reads in fastq format, a library of tranposable elements in fasta format and a reference genome in fasta format. Fasta files should be pre indexed using BWA. This pipeline also requires that BWA and Samtools are installed and on the users $PATH. The following describes the process to extract informative reads from a single sample and will need to be repeated per sample.

Paired end reads are initially aligned to the library of transposable elements manually with bwa mem. The tef-extract-informative tool is then used to identify and extract unmapped reads with mapped mates (informative reads) which are mapped against the reference genome and tagged with the ID of their mates transposable element.

The following file names are used in this example:

- reads1.fastq and reads2.fastq: paired-end reads.
- reference.fasta: reference genome.
- repeats.fasta: reference set of repeat elements.

Initial alignment

Create index fasta files with bwa:

```
bwa index -a bwtsw reference.fasta
bwa index -a is repeats.fasta
```

Map paired end reads to repeat elements using bwa mem and convert to a sorted bam file with samtools:

```
bwa mem -t 8 \
    repeats.fasta \
```

```
    reads1.fastq \
    reads2.fastq \
    | samtools view -Su - \
    | samtools sort - \
    -o reads_mapped_to_repeats.bam
```

Index the resulting bam file with `samtools`:

```
samtools index reads_mapped_to_repeats.bam
```

Extract informative reads

Runing the `tef-extract-informative` program on the indexed bam file:

```
tef-extract-informative \
    reads_mapped_to_repeats.bam \
    --reference reference.fasta \
    --output informative.bam \
    --threads 8
```

Arguments:

- Input bam file
- `-r/--reference` reference genome to align informative reads to.
- `-o/--output` the name of the output (indexed) bam file.
- `-t/--threads` specifies how many threads to use for alignment in bwa (python components of the pipeline are single threaded).

Additional arguments:

- `--exclude-tails` by default the soft-clipped tails of pairs properly mapping to a single repeat element are included as an additional source of information. This option will exclude soft-clipped tails from the resulting bam file.
- `--tail-minimum-length` the minimum length allowed for soft-clipped tails to be included (defaults to `38`).
- `--tempdir` by default, the intermediate files are written to a temporary directory that is automatically removed when the pipeline is completed. There may be limited temporary space on some systems in this option should be used. If this option is used the intermediate files will not be automatically removed.
- `--mate-element-tag` by default, the sam-tag used to store repeat element names associated with each read is `ME` (Mate Element). An alternative tag can be specified with this option.

The output file `informative.bam` contains informative reads mapped to the reference genome. Each of these reads is tagged with the repeat element that their pair was mapped to.

Fingerprinting one or more samples

The `tefingerprint` tool can be used to create a fingerprint of a single sample, or to create a comparative fingerprint of multiple samples (i.e. comparing the fingerprints of more than one sample)

Example usage for comparing two or more bam files:

```
tefingerprint informative1.bam informative2.bam ... \
    -f family1 family2 ... \
    -m 10 \
    -e 350 \
    --gff fingerprint.gff.gz \
    --csv fingerprint.csv.gz
```

Or when specifying most common parameters:

```
tefingerprint informative1.bam informative2.bam ... \
    -a annotation.gff \
    -r chr chr2 chr3 ... \
    -f family1 family2 ... \
    -m 10 \
    -e 350 \
    -b 25 \
    -j 50 \
    -n 3 \
    -q 30 \
    -t 4 \
    --gff fingerprint.gff.gz \
    --csv fingerprint.csv.gz
```

Where `informative1.bam ...` are the bam file(s) being fingerprinted, and `fingerprint.gff.gz` and `fingerprint.csv.gz` are respectively the output in (compressed) csv and gff3 formats (these can be uncompressed by removing the `.gz` extension).

Arguments:

- A single bam file to be fingerprinted or multiple bam files for a comparative fingerprint.
- `-a/--annotation-of-known-elements` An optional annotation of known repeat elements in gff (3) format for matching to identified insertions. Known elements are also used for joining pairs of clusters either side of an insertion. Known elements are also used for joining pairs of clusters either side of an insertion. This gff file may be compressed with gzip or bz2.
- `-r/--references` The reference sequence(s) (e.g. chromosomes) to be fingerprinted. If left blank (None) all references sequences in the input file will be used. *Default = None.*

- `-f/--families` Repeat element/transposon categories to be used. These must be exact string match's to start of read name and are used to split reads into categories for analysis. Not specifying at least one valid category will result in empty output files. *Default = None.*
- `-m/--minimum-reads` Minimum number of read tips required to form a cluster. *Default = 10.*
- `-e/--epsilon` The maximum allowable distance among a set of read tips required to form a cluster. This should be approximately equal to the insert size of paired reads. *Default = 250.*
- `-s/--splitting-method` Method used for splitting proximate clusters. One of "none", "aggressive" or "conservative". See the full documentation for details. *Default = "conservative".*
- `-b/--buffer-fingerprints` Additional buffer to be added to margins of fingerprints. This is used avoid identifying small clusters as unique, when there is only slight miss-match in read positions across samples (i.e. false positives). It also improves the robustness of comparisons by allowing more reads to be included in each bin. The buffer is trimmed back to the extent of the furthermost read tips it contains. *Default = 25.*
- `-j/--join-distance` Used to try and match clusters of informative reads to a known repeat-element (if provided) as well as joining pairs of clusters at either end of a repeat insertion. This represents the maximum distance to search for a known repeat from the end of each cluster. If no know repeat is present (or none are provided) then clusters will be paired if they are within twice this distance of one another. *Default = 25.*
- `-n/--number-of-common-elements` The number of most common repeat elements contributing to each cluster that are counted. *Default = 3.*
- `-q/--mapping-quality` Minimum allowed mapping quality for informative reads mapped to the reference genome. *Default = 30.*
- `-t/--threads` Maximum number of cpu threads to be used. The maximum number of threads that can be utilised is the number of reference molecules to be fingerprinted. *Default = 1.*
- `--gff` File name for GFF output. Compression will be applied by extension e.g. ".gz" or ".bz2". Output may be written to standard output using "-". *Default = None.*
- `--csv` File name for CSV output. Compression will be applied by extension e.g. ".gz" or ".bz2". Output may be written to standard output using "-". *Default = None.*

Additional arguments:

- `--minimum-epsilon` Minimum epsilon values used when calculating support for clusters. This is only used in hierarchical clustering 'and should usually be left as 0. *Default = 0.*
- `--mate-element-tag` Sam-tag used in bam file to indicate the repeat element matched to each the mate read. *Default = "ME".*
- `--no-colour` Switch to disable colour coding of gff output. This may improve performance

Filtering GFF Output

The `tef-filter-gff` script can be used to filter down the gff formatted results of `tefingerprint` . Filters can be applied to attributes in the attribute column or to the first 8 standard gff3 columns. The first 8 standard gff3 columns are respectively named "seqid", "source", "type", "start", "end", "score", "strand" and "phase".

Filters take the form `'<column/attribute><operator><value>'` where:

- `<column/attribute>` is the name of the column or attribute that the filter is applied to.
- `<operator>` is one of the following operators `=`, `==`, `!=`, `< >`, `>=`, `<=` that describes the comparason being performed.
- `<value>` is the value the each feature is compared to.

Filters should be contained within quotes `''` so that the operator is not interpreted as a shell command.

The following operators are only used for numerical comparisons: `< >`, `>=`, `<=`.

The operators `=`, `==` and `!=` will try to compare values as numerical (floating points) but will also check for equivalence or non-equivalence of string values. Note that `=`, `==` are identical.

Multiple filters may be combined within an "all" or "any" context. I.e. in an "all" context each feature must match all of the filters to be kept. In an "any" context each feature must only one of the filters to be kept. If both an "all" and an "any" context ar used then they are evaluated separately before being combined in an aditional "all" context.

Unix style wildcards may be used and will expand to match all possible column and attribute fields that they can. The resulting set of filters will then be evaluated within the context of the original filter.

Example usage with one column filter and two attribute filters:

```
tef-filter-gff fingerprint.gff.gz \
    --all 'seqid=chr1' 'start>=1000' 'stop<9000' \
    --any 'sample_?_count>100' \
    -o fingerprint_filtered.gff.gz
```

Where `fingerprint.gff.gz` is a gff file compressed with gzip and `fingerprint_filtered.gff.gz` is a filtered version of that file.

The above example is evaluated as follows: the "all" context will select only feature from chromosome 1 that are in the interval 1000-8999. The "any" context contains a filter with the wildcard "?" which will expand the filter to match multiple samples and evaluate each of the resulting filters e.g.: with three samples it would expand to the equivalent of `--any 'sample_0_count>100' 'sample_1_count>100' 'sample_2_count>100'`. Therefore the full command

would select features where any one of the samples contains more than 100 reads, from within the interval chr:1000-8999.

Arguments:

- A gff (3) file to be filtered. If `-` is specified then gff lines will be read (in plain text) from standard in.
- `--all` filters to apply to apply in the "all" context. These should take the form `'<column><operator><value>'`
- `--any` filters to apply to apply in the "any" context. These should take the form `'<column><operator><value>'`
- `-o/--output` a file to write the data to. By default `-` the data are written to standard out. If the data are written to a file, this file will be compressed with gzip or bz2 based on the files extension e.g. `.gff.gz` or `.gff.bz2`.