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
```

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`.

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.

You should have the following four files:

- `reads1.fastq` and `reads2.fastq`: paired-end reads in fastq format.
- `reference.fasta` a reference genome in fasta format.
- `repeats.fasta` a library of repeat elements in fasta format.

Initial alignment

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. These files can be saved by manually specifying a directory with theis option.
- `--mate-element-tag` by default, the same tag used to store repeat element names associated with each read is `ME` (Mate Element). This can be changed with the 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 bam files:

```
tefingerprint informative1.bam informative2.bam ... \
    -a annotation.gff \
    -f family1 family2 ... \
    -m 20 \
    -e 500 \
    -b 50 \
    -j 50 \
    -n 3 \
    -q 30 \
    -t 4 \
    --gff fingerprint.gff \
    --csv fingerprint.csv
```

Where `danglers.bam` is the bam file being fingerprinted and `fingerprint.gff` is the output gff file.

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 elements in gff format for matching to identified insertions. Known elements are also used for joining paris of clusters either side of an insertion.
- `-r/--references` may optionally be used to specify a subset of chromosomes to fingerprint. By default all reference chromosomes are fingerprinted (based on the bam header).
- `-f/--families` specifies the (super) families or grouping of repeated elements to fingerprint. These names are matched against the start of the mate element name i.e. the name `Gypsy` would treat reads with tagged with a mate element called `Gypsy3`, `Gypsy27` or `GypsyX` as the same.
- `-m/--minimum-reads` specifies the minimum number of read (tips) required to form a cluster.
- `-e/--epsilon` specifies the maximum allowable distance among a set of read tips to be considered a cluster.
- `-b/--buffer-fingerprints` specifies a distance (in base pairs) to buffer fingerprints by before counting reads (defaults to `0`). This is used to ensure that small clusters, that are slightly offset in different samples, are treated as a single comparative bin. It also improves the robustness of comparisons by allowing more reads to be included in each bin.
- `-j/--join-distance` used to try and match clusters of informative reads to a known transposon (if provided) as well as joining pairs of clusters at either end of a transposon insertion. This value represents the maximum distance to search for a known transposon and half the maximum distance to search for a paired cluster if no a known transposon is identified (defaults to `0`).
- `-n/--number-of-common-elements` The number of most common elements contributing to each cluster that are counted. I.e. if this value is 3 then for each cluster of each sample the name and counts of the three most common elements are recorded (defaults to `3`).
- `-q/--mapping-quality` specifies the minimum mapping quality allowed for reads (defaults to `30`).
- `-t/--threads` specifies the number of CPU threads to use. The maximum number of threads that may be used is the same as the number of references specified.
- `--gff` create a gff file of the resulting data. This data can be sent to standard output for piping using `--gff -`
- `--csv` create a csv file of the resulting data. This data can be sent to standard output for piping using `--csv -`

Additional arguments:

- `--minimum-epsilon` the minimum value of epsilon to be used in hierarchical clustering (defaults to `0`).

- `--non-hierarchical` by default a hierarchical clustering algorithm is used. This flag will switch to the non-hierarchical version.
- `--mate-element-tag` the sam tag used to specify the name of each reads mate element (defaults to `ME`).

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

Where `fingerprint.gff` is a gff file and `fingerprint_filtered.gff` 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 equivilent 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:

- `--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>'`