TEiso: a pipeline for identification of transcripts isoforms in the context of transposable elements

TEiso is a pipeline that can analyze RNA-seq data and find the relation between TSS of each trancscripts (isoforms) and the closest transposable elements (TEs). This tutorial will describe how users can run TEiso in shell command.

Dependencies

Bowtie, Tophat, Cufflinks, Cuffcompare, Bedtools_closest.

Installation

$tar -xzf TEiso-1.2.tar.gz

$cd TEiso-1.2/

$python setup_TEiso.py install

$export PYTHONPATH=$PWD

$export PATH=$PATH:$PWD/bin

Options

-f	**INPUT_REFERENCE**	**ref sequences**
-g	**INPUT_TRANSCRIPTS**	**INPUT_TRANSCRIPTS GTF/GFF with known transcripts**
-e	**SINGLE_PAIRED**	**type of input reads, single or paired end**
-s	**SINGLE_READ**	**a single input read**
-l	**LEFT_READ**	**left reads**
-r	**RIGTH_READ**	**right reads**
-t	**INPUT_TE**	**GFF with known transposable_element**
-a	**ASSEMBLY_TOOL**	**type of RNA-Seq assembly tool**

This tutorial will explain the basics analysis used in TEiso, and will include the following:

For more details on data format, see the slideshow « Introduction_formats_sequences » on URGI download tutorial repository.

1) Indexing Sequences and Mapping

TopHat is a program that aligns RNA-Seq reads to a genome in order to identify exon-exon splice junctions. To make mapping faster, The reference genome must first be "indexed" by bowtie-build. For more information see bowtie-bio.sourceforge.net manual and ccb.jhu.edu_tophat manual.

Input format

Bowtie-build takes the sequence(s) of genome to build to build indexing sequences. Please note that the index files must be keeped in the same directory of mapping.

Tophat takes reads files in Sanger FASTQ format and a FASTA file with the sequence(s) of genome.

Outputs

Bowtie-build produces a set of 6 files with suffixes .1.ebwt, .2.ebwt, .3.ebwt, .4.ebwt, .rev.1.ebwt, and .rev.2.ebwt. (If the total length of all the input sequences is greater than about 4 billion, then the index files will end in ebwtl instead of ebwt.) These files together constitute the indexwhich are needed to align reads to that reference. These results are in folder « bowtie_build ».

Tophat script produces a number of intermediate files which are in folder « tophat ».

We used the default values for all options of bowtie-build and tophat.

2) Transcriptome assembly with Cufflinks

Cufflinks the program assembles transcriptomes from RNA-Seq data and quantifies their expression. One of Cufflinks' best features is that it can function as a reference-based transcriptome assembler. Cufflinks can use reference transcript annotation to guide assembly by using option *-g*. It can also identify novel transcripts in your sequencing data by examining their alignments to the genome. Here, we use this option (*-g)* to generate the novel transcripts . For more information see cole-trapnell-lab.github.io cufflinks tool documentation.

Input format

Cufflinks takes a list of read alignments in SAM or BAM format. In this pipeline, we use «accepted_hits.bam» which is generated by Tophat.

We used the default values for all options of Cufflinks.

Outputs

Cufflinks produces several output files which are in folder « cufflinks ».

3) Comparing the structure of assembled transcripts with Cuffcompare

After assembling a transcriptome from one or more samples, we compare our assembly to known transcripts. Cuffcompare examines the structure of each isoforms (here, created transcript by cufflinks against a reference transcriptome) and then it generates a class code for each of them. For more information see broadinstitute.org documentation on Cufflinks.cuffcompare.

Class Codes

If you ran cuffcompare with the -r option, tracking rows will contain the following values.

```
Priority     Code      Description
---------------------------------
1            =            Match
2            c            Contained
3            j            New isoform
4            e            A single exon transcript overlapping a reference
                       exon and at least 10 bp of a reference intron,
                       indicating a possible pre-mRNA fragment.
5            i            A single exon transcript falling entirely with a
                       reference intron
6            r            Repeat. Currently determined by looking at the
                          reference sequence and applied to transcripts
                          where at least 50% of the bases are lower case
7            p            Possible polymerase run-on fragment
8            u            Unknown, intergenic transcript
9            o            Unknown, generic overlap with reference
10           .            (.tracking file only, indicates multiple classifications)
```

Input format

Cuffcompare takes a GTF file (here produced by Cufflinks) and a "reference" annotation.

We used the default values for all options of Cuffcompare.

Outputs

Cufflinks produces four output files which are in folder « cuffcompare ».

4) Converting transcriptome assembly to bed file

CufflinksGTFToBed is a python script to convert the transcriptome assembly of cufflinks (gtf file) to a bed file. This step parses a gtf file to extract all essential information for this pipeline and then writes output as bed file. Tracking columns of bed file will contain the following values.

```
Column      Description
--------------------------------
1           Name of the chromosome or scaffold of the transcript
2           Start position of the transcript
3           End position of the transcript
4           ID of the transcript
5           ID of the gene associated with the transcript
6           Strand of the transcript (defined as + (forward) or - (reverse).)
7           calculated FPKM of the transcript by Cufflinks
```

Input format

CufflinksGTFToBed takes a GTF file (here produced by Cufflinks).

Outputs

CufflinksGTFToBed produces a bed files which is in « bedtools_closest » .

5) Converting transposable elements annotation file to bed file

GFFToBed is a python script to convert a gff file of the transposable elements (TEs) to a bed file. This step parses a gff file to extract all essential information for this pipeline and then writes output as bed file. Tracking columns of bed file will contain the following values.

```
Column      Description
--------------------------------
1           Name of the chromosome or scaffold of the TE
```

```
2           Start position of the TE
3           End position of the TE
4           ID of the TE
5           Target of the TE (here result of REPET)
6           Strand of the TE (defined as + (forward) or – (reverse).)
```

Input format

GFFToBed takes a GFF file (here produced by Cufflinks).

Outputs

GFFToBed produces a bed files which is in folder « bedtools_closest ».

6) Taking the closest transposable elements to transcriptomes with bedtools

Bedtools closest is a tool to find the closest locations between two bed files. TEiso goal is to find the nearest transposable elements (TEs) to each transcriptomes by Bedtools closest. As we explained above, two python scripts CufflinksGTFToBed and GFFToBed creat the bed files with corresponding to transcriptome assembly of cufflinks (step 4) and transposable elements (step 5) . Here, bedtools closest uses these two bed files to find the clossest TEs to each transcriptome (isoforms) and also calculates the distance between them. The output of bedtools closest is also into bed file. Tracking columns of final bed file will contain the following values. For more information see bedtools.readthedocs.io latest documentation on closest.

```
Column      Description
--------------------------------
1           Name of the chromosome or scaffold of the transcript
2           Start position of the transcript
3           End position of the transcript
4           ID of the transcript
5           ID of the gene associated with the transcript
6           Strand of the transcript (defined as + (forward) or - (reverse).)
7           calculated FPKM of the transcript by Cufflinks
8           name of the chromosome or scaffold of the closest TE
9           Start position of the closest TE
10          End position of the closest TE
11          ID of the closest TE
12          Target of the closest TE (here result of REPET)
13          Strand of the closest TE (defined as + (forward) or – (reverse).)
14          Distance between closest TE and transcript.The reported distance for
overlapping/including features will be 0.
```

Input format

Bedtools closest takes two BED files. In this pipeline, the first bed file is for transcriptomes (produced by CufflinksGTFToBed) and the second one is for TEs (produced by GFFToBed).

We used the default values for all options of Bedtools closest.

Outputs

Bedtools closest produces a output file « bedtools_closest_cufflinks_transcripts » in format Bed in folder « bedtools_closest ».

7) Finding cases where TSS are closest to TEs

TEiso goal is to find the nearest transposable elements (TEs) to the TSS of each transcriptomes (isoforms). ClosestToStartSite is a python script to parses the results of bedtools closest (step 6) and extract the cases where TEs are close/overlap to/withTSS of each isoforms. The output of this step is a bed file. Tracking columns of final bed file will contain the following values.

```
Column      Description
--------------------------------
1           Name of the chromosome or scaffold of the transcript
2           Start position of the transcript
3           End position of the transcript
4           ID of the transcript
5           ID of the gene associated with the transcript
6           Strand of the transcript (defined as + (forward) or - (reverse).)
7           calculated FPKM of the transcript by Cufflinks
8           name of the chromosome or scaffold of the closest TE
9           Start position of the closest TE
10          End position of the closest TE
11          ID of the closest TE
12          Target of the closest TE (here result of REPET)
13          Strand of the closest TE (defined as + (forward) or – (reverse).)
14          Distance between closest TE and transcript.
            For Case TE overlap transcript: distance is region overlap
15          Discription of the TE's position according to the TSS
            TE_near_TSS ,  TE_overlap_TSS, TE-inclus-gene, gene-inclus-TE
```

Input format

ClosestToStartSite takes a BED file which is produced by Bedtools closest (bedtools_closest_ cufflinks_transcripts)

Outputs

ClosestToStartSite produces an output file in format Bed « bedtools_closest_cufflinks_transcripts_ TSSoverlaps_and_TE_near_TSS_and_inclus_ALL_with_Ref » in folder « bedtools_closest ».