TSSer manual

Hadi Jorjani

Email : hadi.jorjani@unibas.ch

October 25, 2013

Contents

1 Introduction

TSSer is a computational pipeline for identifying transcription start sites (TSSs) based on differential RNA-seq data. It relies on quantification of the enrichment of reads starting at a particular position in a sample treated with 5' mono-phosphate-dependent terminator exonuclease (TEX) relative to an untreated sample and also on the significance of the reads starting at the specific position compared to their nearby positions. *TSSer* can further take replicates into account, computing an overall probability that the expression from a given genomic position is enriched and treated compared to untreated samples.

The input to TSSer consists of replicates of paired samples (enriched and non-enriched libraries) in "bedweight" format (a special case of BED format where the score field holds the copy number of the read) which is described below.

2 Quick Start

TSSer can be downloaded from : http://www.clipz.unibas.ch/TSSer/index.php

After downloading, the files can be unzipped as follows:

```
unzip TSSer.zip
```

This creates the "TSSer" directory in which all the scripts that are used in the pipeline can be found. Example "input files" are also provided and can be downloaded from the above link, then uncompressed as follows:

```
unzip input_example.zip
```

which leads to "my_data_example" directory containing example input files, description of the file format and contents.

There are two variants of the main script for scoring putative TSSs. The first is a fast shell script, "main_pipe_zscore.sh", which calculates the probability of 5' enrichment based on the z-score. The second, "main_pipe_lambda.sh", implements the "lambda-score" described in the manuscript and is somewhat slower because it calculates an integral. The Python libraries "numpy", "scipy" and "sympy" are required to be installed.

2.1 Input File Format

Example input files are provided in the "my_data_example" directory. They should be:

1. Files with the names of replicate samples:
   They just have to have uniq names. For instance,

   ```
   rep_1
   rep_2
   rep_3
   ```

   For each sample it is assumed that a file with TEX-enriched samples data and one with not-enriched samples data are provided. The names of these files will be obtained by appending to the sample name the suffix "+" or "-" followed by ".bedweight".

2. The annotation of genome of interest is used to set the thresholds for calling TSSs, based on the accuracy of identification of TSSs for known genes. Thus, the protein table of the organism in "ptt" format (as an example: "NC_016810.ptt") needs to be downloaded from NCBI, in the case of our examples from "NCBI":

3. Reads corresponding to structural RNAs such as ribosomal RNAs are not used in the annotation process. The structural RNA table of the organism in "rnt" format also needs to be downloaded from NCBI. In the case of our examples the file "NC_016810.rnt" has to be downloaded from "NCBI":

4. Data from the pair of enriched and not-enriched samples for each replicate in bedweight format. The "bedweight" format is similar to "BED" format except for the fifth column (score field) which in the bedweight file contains the copy number of the read. Thus, the fields in the files are:

   - reference genome ID
   - read start
   - read end
   - read ID
   - copy number
   - strand

   Here is an example of bedweight format:

   ```
   gi|378697983|ref|NC_016810.1|   3425415 3425430 seq_896 0.5     -
   gi|378697983|ref|NC_016810.1|   1068982 1068997 seq_896 0.5     -
   gi|378697983|ref|NC_016810.1|   2080956 2080992 seq_897 1.0     +
   gi|378697983|ref|NC_016810.1|   3162669 3162695 seq_898 0.3     +
   gi|378697983|ref|NC_016810.1|   3162563 3162589 seq_898 0.3     +
   gi|378697983|ref|NC_016810.1|   3479220 3479247 seq_898 0.3     -
   gi|378697983|ref|NC_016810.1|   2418324 2418437 seq_899 5.0     +
   gi|378697983|ref|NC_016810.1|   1893702 1893816 seq_900 15.0    -
   gi|378697983|ref|NC_016810.1|   3034111 3034225 seq_901 1.0     -
   ```

Description of "BED" format is given at *https://genome.ucsc.edu/FAQ/FAQformat.html#format1*:

Programs for aligning short reads to genomes (such as *BWA*[1], *Bowtie*[2] and *Segemehl*[3] usually produce files in the "SAM" format which can be converted to "BED" format using "BEDtools" package that can be obtained from *http://code.google.com/p/bedtools/downloads/list*. The "samtools" package can be obtained from *http://sourceforge.net/projects/samtools/*. For example:

```
#convert sam to bam
~/samtools-0.1.18/samtools view -bS my_sample.sam > my_sample.bam
#convert bam to bed
~/BEDTools-Version-2.16.2/bin/bamToBed -i my_sample.bam  > my_sample.bed
```

To convert the "BED" format to "bedweight" format we provide a Python script in the "TSSer" directory which can be applied as follows:

```
#convert bed to bedweight
./bed_to_bedweight.py my_sample.bed > my_sample.bedweight
```

2.2 Running the Pipeline

The steps of the pipeline are specified in the shell script "main_pipe_lambda.sh" or "main_pipe_zscore.sh". Also a slim summary of different segments of the pipeline and corresponding output of each segment is provided in "pipeline_outline.txt" file. Before running the main pipeline script i.e. "main_pipe_zscore.sh" or "main_pipe_lambda.sh", First you have to edit them and initialize variable names (e.g. main_dir, genome_ID, normalization_factor , ...) to their appropriate values.

```
my_data_dir=~/Pipeline/my_data/
main_dir=~/Pipeline/
Programs_dir=$main_dir/TSSer/
result_dir=$main_dir/Output/
genome_ID=NC_016810
normalization_factor=100000
```

Then you can simply run the script as shown below:

```
sh ./main_pipe_zscore.sh
```

or

```
sh ./main_pipe_lambda.sh
```

A brief description of each step and its output files is included in the comments. The output files are usually tab-delimited files and the description of the columns is given between brackets, below the output file name. For example:

```
## finding nearest annotated gene for each start site
$Programs_dir/find_nearestGene.sh
## output: "sample_name_P_nearestGene" and "sample_name_N_nearestGene"
## output format: [start_site, gene_name, distance to start codon, CDS start]
```

3 Output

Running the pipeline creates a log file in the working directory which its defalut name is "*./TSSer_log*" which makes it possible to track the status of pipeline in case of occuring an error. The pipeline is composed of 15 submodules which are denoted by latin numbers, and after finishing each module, the intermediate outputs are generated in the output directory and will be written in the log file. The final result is a list of selected TSSs and their properties (e.g. probability of 5' enrichment, enrichment relative to local neighborhood, z_score, normalized expression, nearest annotated gene, classification of the TSS, ...). Additional intermediate outputs are available. They are described in the pipeline main script, at the places where they are generated.

References

[1] Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform**. *Bioinformatics* 2009, **25**(14):1754–1760.

[2] Langmead B, Trapnell C, Pop M, Salzberg SL: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome**. *Genome Biology* 2009, **10**:R25.

[3] Hoffmann S, Otto C, Kurtz S, Sharma CM, Khaitovich P, Vogel J, Stadler PF, Hackermuller J: **Fast Mapping of Short Sequences with Mismatches, Insertions and Deletions Using Index Structures**. *PLoS Comput Biol* 2009, **5**:e1000502.