Fast quantification of splice junctions by *sjcount*

Dmitri D. Pervouchine

September 30, 2013

1 Synopsis

The purpose of *sjcount* is to provide a fast method for quantification of splice junctions from BAM files. It is an annotation-agnostic version of bam2ssj.

2 Installation and usage

See README.md file for installation instructions. *sjcount* is used with the following keys

```
sjcount -bam bam_file [-ssj junctions_output] [-ssc boundary_output]
       [-maxlen max_intron_length] [-minlen min_intron_length] [-margin length]
       [-read1 0|1] [-read2 0|1] [-nbins number_of_bins] [-binsize bin_size]
       [-lim number_of_lines] [-quiet]
```

where

- **bam_file** is a sorted input BAM file with a header
- **junctions_output** is the output file with junction counts
- **boundary_output** is the output file with boundary counts
- **maxlen** upper limit on intron length, 0 = no limit (default=0)
- **minlen** lower limit on intron length, 0 = no limit (default=0)
- **margin** length, see below, (default=0)
- **read1** 0/1, reverse complement read1 no/yes (default=no)
- **read2** 0/1, reverse complement read2 no/yes (default=no)
- **binsize** size of the overhang bin, (default=∞)
- **nbins** number of overhang bins, (default=1)
- **lim** nreads stop after nreads, (default=no limit)

- **quiet** – suppress verbose output

The output consists of two parts. First, a tab-delimited file containing splice junction counts is produced. Its format is as follows

```
chr1    100 200 -1 10 25
chr1    100     200     -1      11      12
... ... ... ... ... ...
```

where the first column contains chromosome id, the second and the third columns contain positions of terminal exonic nucleotides which define the splice junction, the fourth column contains strand (1 or -1), the fifth column is the overhang (see definitions below), and the last column is the respective number of reads with these properties.

The secons output is a tab-delimited file which contains counts of continuous (non-split reads) which *overlap* splice sites of splice junctions tabulated in the previous step. This second file is optional and is used to compute the completness of splicing index [2, 3].

3 Definitions

By definition, we say that we observe a splice junction each we see an 'N' symbol in the CIGAR attribute of some SAM alignment. For instance, the alignment shown in Figure 1 below gives rise to two splice junctions, denoted by SJ_1 and SJ_2. We keep the convention that coordinates of splice junctions always refer

```
       10        20        30        40        50        60        70        80
       |         |         |         |         |         |         |         |
       12345678 90123456789012345678901234567890123456789012345678901234567890123456789012

Ref    AGTCTAGG*GACGGCATAGGAGGTGAGCATTTGTGTACGCAGATCTACAAAACATGTGTCACGGATAGGATCG
Query     CTAGGAGACGG**TAGGAG.....................ATCTA*AAAACAT.............GATa
                           |<-----   SJ1  ----->|           |<--- SJ2 --->|
```

The corresponding SAM line is:

```
Query   123   Ref   14    255    5M1I5M2D6M20N5M1D7M13N3M1S 1234
```

Figure 1: An example alignment and its CIGAR attribute

to terminal exonic nucleotides, i.e., SJ_1 is Ref_31_52 and SJ 2 is Ref_64_78. We also denote the length of the intron by $l(\mathrm{SJ})$, i.e. $l(\mathrm{SJ}_1) = 52 - 31 - 1 = 20$ and $l(\mathrm{SJ}_2) = 78 - 64 - 1 = 13$. Intron length is always equal to the corresponding 'N' number in the CIGAR attribute.

Each splice junctions is associated with four numbers: m_u (m_d) — the number of matching nucleotides immediately upstream (downstream) of the junction, and v_u (v_d) — the length in the reference of the aligned region, also called overhang, which includes M/I/D CIGAR operations and is located immediately

upstream (downstream) of the junction. In Figure 1 we have $m_u(\mathrm{SJ}_1) = 6$, $m_d(\mathrm{SJ}_1) = 5$, $v_u(\mathrm{SJ}_1) = 31 - 14 + 1 = 18$, $v_d(\mathrm{SJ}_1) = 64 - 52 + 1 = 13$ and $m_u(\mathrm{SJ}_2) = 7$, $m_d(\mathrm{SJ}_2) = 3$, $v_u(\mathrm{SJ}_2) = 64-52+1 = 13$, $v_d(\mathrm{SJ}_2) = 80-78+1 = 3$.

For each splice junction we require that

1. $l(\mathrm{SJ}) \geq$ **minlen** and $l(\mathrm{SJ}) \leq$ **maxlen**
2. $m_u \geq$ **margin** and $m_d \geq$ **margin**

In addition to junction coordinates, the overhang $v_u(\mathrm{SJ})$ can also be used to correct for artifactually large read counts that arise in certain positions [1]. In Figure 2 we show six split reads supporting the same splice junction with overhangs 14 (Q1), 12 (Q2–Q4), and 8 (Q5–Q6).

```
       10        20        30        40        50        60        70        80
       |         |         |         |         |         |         |         |
       12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

Ref    AGTCTAGGGACGGCATAGGAGGTGAGCATTTGTGTACGCAGATCTACAAAACATGTGTCACGGATAGGATCG

Q1           GGACGGCATAGGAG...................ATCT
Q2             ACGGCATAGGAG...................ATCTAC
Q3             ACGGCATAGGAG...................ATCTAC
Q4             ACGGCATAGGAG...................ATCTAC
Q5                 CATAGGAG...................ATCTACAAAA
Q6                 CATAGGAG...................ATCTACAAAA
```

Figure 2: Split reads support the same splice junction with different overhangs

The quantification of abundance is done as follows. For each splice junction (pair of coordinates) we initialize and keep *nbins* separate counters. For each instance of a splice junction we increment the counter corresponding to the overhang bin defined by $d = floor(v_u/binsize)$.

For example, in the default settings we have $binsize = +\infty$. This means that $d = 0$ for all supporting reads, regardless of their overhang ($v_u = 14$ for Q1, $v_u = 12$ for Q2–4, and $v_u = 8$ for Q5–6 in Figure 2). Therefore, there is only one counter to increment, and the result will be the "collapsed" counts. The output corresponding to Figure 2 will then be

```
Ref     31      52      1       0       6
```

By contrast, to take into account the overhang information, one should set $binsize = 1$ (and also specify *nbins* because the program doesn't know the range of possible overhang values). There will be a separate counter for each offset d and the output corresponding to Figure 2 will be

```
Ref     31      52      1       8       2
Ref     31      52      1       12      3
Ref     31      52      1       14      1
```

Note that when aggregated by the fifth column, the number of counts coincides with the collapsed counts.

References

[1] B. Kakaradov, H. Y. Xiong, L. J. Lee, N. Jojic, and B. J. Frey. Challenges in estimating percent inclusion of alternatively spliced junctions from RNA-seq data. *BMC Bioinformatics*, 13 Suppl 6:S11, 2012.

[2] D. D. Pervouchine, D. G. Knowles, and R. Guigo. Intron-centric estimation of alternative splicing from RNA-seq data. *Bioinformatics*, 29(2):273–274, Jan 2013.

[3] H. Tilgner, D. G. Knowles, R. Johnson, C. A. Davis, S. Chakrabortty, S. Djebali, J. Curado, M. Snyder, T. R. Gingeras, and R. Guigo. Deep sequencing of subcellular RNA fractions shows splicing to be predominantly co-transcriptional in the human genome but inefficient for lncRNAs. *Genome Res.*, 22(9):1616–1625, Sep 2012.