Fast quantification of splice junctions from RNA-seq data by *sjcount* v2.0

Dmitri D. Pervouchine[*]

Centre for Genomic Regulation (CRG), Barcelona, Spain

April 1, 2014

1 Synopsis

The purpose of *sjcount* is to provide a fast method for quantification of splice junctions from BAM files. It is the annotation-agnostic version of bam2ssj. This document describes the version **v2.0** of *sjcount*. The older version of *sjcount* (v1.0) is also included in the package inder the name *sjcount-deprecated*.

2 Changes w.r.t to the previous version (v1.0)

Below are the major changes that make v2.0 different from v1.0.

1. The computation of reads overlapping exon boundaries now includes all, not only continuous reads
2. -maxlen upper limit on intron length: deprecated
3. -minlen lower limit on intron length: deprecated
4. -margin, minimum number of flanking nucleotides to support SJ or EB: deprecated
5. -binsize, the size of bins for offsets: deprecated
6. -maxnh the max value of the NH tag (set to 1 for uniquely mapped reads): added

[*]email: dp@crg.eu

3 Installation and usage

See README.md file for installation instructions. The program *sjcount* is used from the command line with the following keys

```
sjcount -bam bam_file [-ssj junctions_output] [-ssc boundary_output]
       [-read1 0|1] [-read2 0|1] [-unstranded] [-nbins number_of_bins]
       [-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
- **read1** 0/1, reverse complement read1 no/yes (default=no)
- **read2** 0/1, reverse complement read2 no/yes (default=no)
- **unstranded**, force strand=0
- **nbins** number of offset bins, (default=1)
- **maxnh** the max value of the NH tag, (default=none)
- **lim** stop after reading these many lines, (default=no limit)
- **quiet** – suppress verbose output **NOTE: use -quiet if you redirect stderr to a file!**

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

```
chr1    100     200     +       10      25
chr1    100     200     +       11      12
... ... ... ... ... ...
```

where the first column contains chromosome, the second and the third columns contain the positions of *terminal exonic nucleotides* corresponding to a splice junction, the fourth column contains strand ('+' or '-' for stranded or '.' for unstranded data), the fifth column is the offset (defined below), and the last column is the respective count, i.e., the number of split-mapped reads with the given combination of chromosome, begin, end, strand, and offset.

The secons output file is also a tab-delimited file which contains read counts of read alignments which *overlap* exon boundries (exon boundries are defined by splice junctions in the previous file). In this version all alignments that overlap an exon boundary by at least one nucleotide are counted (in older versions only continuous alignments were considered. This second file is optional and is needed to compute the completness of splicing index [2, 3].

4 Definitions

By definition, we say that we observe a *splice junction* each time we see an 'N' symbol in the CIGAR attribute of the alignment. If a CIGAR attribute contains several N's, it will be counted several times. Splice junctions are decided entirely by the mapper which produced the alignment. Each splice junction is characterized by a combination of four attributes: chromosome, start, end, and strand. We keep the convention that start and end of a splice junction always refer to the terminal *exonic* nucleotides. For instance, the alignment shown in Figure 1 below corresponds to two splice junctions, denoted by SJ_1 and SJ_2. The

```
       10        20        30        40        50        60        70        80
       |         |         |         |         |         |         |         |
       12345678 9012345678901234567890123456789012345678901234567890123456789012

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

coordinates of these splice junctions are $\mathrm{SJ}_1 = \mathrm{Ref\_31\_52}$ and $\mathrm{SJ}\,2 = \mathrm{Ref\_64\_78}$. Denote by $l(\mathrm{SJ})$ the length of the spliced region, i.e. $l(\mathrm{SJ}_1) = 52 - 31 - 1 = 20$ and $l(\mathrm{SJ}_2) = 78 - 64 - 1 = 13$. Note that $l(\mathrm{SJ})$ is equal to the corresponding 'N' number in the CIGAR attribute.

Artifacts may arise from combining counts that come from different starting positions of the alignment. We define *offset* $t(\mathrm{SJ})$ to be the distance (*in the query sequence!*) from the first alignment position to the corresponding 'N'. For instance, $t(\mathrm{SJ}_1) = 17$ and $t(\mathrm{SJ}_2) = 29$. Since offset is defined as a refernce in the query sequence, its values are bounded by the read length.

Some offsets may give rise to artifactually large read counts [1]. In Figure 2 we show six split reads supporting the same splice junction with offsets 14 (Q1), 12 (Q2–Q4), and 8 (Q5–Q6). Although it might seem controversial, offsets appear decreasing when sequentially processing lines a sorted BAM file.

```
        10        20          30         40          50         60         70          80
        |         |           |          |           |          |          |           |
        1234567890123456789012345678901234567890123456789012345678901234567890123456789012

Ref     AGTCTAGGGACGGCATAGGAGGTGAGCATTTGTGTACGCAGATCTACAAAACATGTGTCACGGATAGGATCG

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

Figure 2: Split-mapped reads support the same splice junction with different offsets

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 its offset. If the offset is larger than or equal to *nbins* then it is set to be equal to $nbins - 1$.

For example, in the default settings we have $nbins = 1$. This means that the bin number will be $1 - 1 = 0$ for all supporting reads, regardless of their offset ($t = 14$ for Q1, $t = 12$ for Q2–Q4, and $t = 8$ for Q5–Q6 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      +       0       6
```

By contrast, if we set $nbins = 1$, there will be a separate counter for each offset and the output corresponding to Figure 2 will be

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

Note that when aggregated by offset with the aggregation function $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$, the result coincides with the collapsed number of counts; for $f(x_1, \ldots, x_n) = \theta(x_1) + \cdots + \theta(x_n)$, where $\theta(x) = 1$ for $x > 0$ and $\theta(x) = 0$ for

$x \leq 0$, the result is the number of *staggered* counts. Also

$$f(x_1, \ldots, x_n) = \log_2(\sum_{i=1}^{n} x_i) - \frac{\sum_{i=1}^{n} x_i \log_2(x_i)}{\sum_{i=1}^{n} x_i}$$

gives entropy of the distribution, which can be used to filter out non-uniform distiburtion of read 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.