JAGuaR

Junction Alignments to Genome for RNA-seq Reads

Documentation

JAGuaR is an alignment protocol for paired end RNA-seq reads based on an extended reference. It uses BWA to align reads to the genome and reference transcript models (including annotated exon-exon junctions) specifically allowing the possibility for a single read to span multiple exons. Reads aligned to the special reference repository are then "repositioned" on to genomic coordinates, transforming reads spanning multiple exons into large-gapped alignments.

Availability: JAGuaR is implemented in Python, and is freely available for download at

http://www.bcgsc.ca/platform/bioinfo/software/jaguar

JAGuaR creates a genome + junctions reference to be used for different read length alignments. JAGuaR has been tested with alignments using BWA (http://bio-bwa.sourceforge.net/). After reads are aligned to this reference with BWA, convertJunctions converts the junction aligned reads to genomic coordinates.

A) Pipeline

1. Create tab delimited transcript annotation file in format which is used to create the junction reference where exon start and end is 1-based inclusive:

   ```
   transcript_id chr exon_start exon_end exon_rank strand
   ```

   This format can be easily generated using BioMart by choosing 'structures' and then clicking the appropriate categories in the GENE and EXON section in the order you want them to appear from left to right. http://www.biomart.org/biomart/martview/

   See Section C for an example of a formatted annotation file.

2. Run ***createJunctions***

   Creates the exon-exon junction sequences for each chromosome and an index with the stub chrX_JN. Concatenates the files created in step2, with the reference genome fasta (i.e. hg19.fa) file to create the genome + junction reference sequence (i.e ref.fa) for alignment.

   This has to be run once for each read size such as 50bp, 75bp, 100bp to create ref.fa.

   ```
   python createJunctions.py -c junction.cfg
   ```

May, 2014

The reference file containing exon-exon junction sequence needs to be indexed before alignment.

```
bwa index -a bwtsw ref.fa
```

See Section B for an example configuration file.

Note:
The chromosome naming used in the transcript annotation file should match the naming used in the genome fasta. If the annotation includes chromosomes not included in the genomic fasta, a message of the form "INFO Chromosome <$chrName> is not included in the genomic fasta" will appear in the createJunctions.log file. In addition the version of BWA used to index the reference should be the same version used for the initial alignment.

3. **Alignment**

Alignments of reads to the JAGuaR reference has only been tested with BWA in the following pipeline:

```
bwa index -a bwtsw ref.fa          [only needs to be run once/genome]

bwa aln ref.fa read1.fastq > read1.sai
bwa aln ref.fa read2.fastq > read2.sai

bwa sampe -s ref.fa read1.sai read2.sai read1.fastq read2.fastq >
aln.sam
```

JAGuaR has been tested with the BWA-MEM algorithm and while some development is required for JAGuaR to make full use of the BWA-MEM algorithm, running in its current version gives similar results for BWA but overall runtime is much faster.

```
bwa-0.7.4/bwa mem -t 8 ref.fa read1.fastq read2.fastq > aln.sam
```

For runs with BWA-MEM only, a post alignment script provided in the JAGuaR package needs to be run before convertJunctions. This chooses the best read when a read is multi-mapped or split.

```
python samFilterSelectReads.py aln.sam > aln.filtered.sam
```

4. Run ***convertJunctions*** to create a repositioned sam file that is sorted by read name:

```
python convertJunctions.py --samtools samtools –f aln.sam –o
myaln -m 34622 -i ref100/ -s
```

To create a repositioned bam file sorted by coordinate and indexed:

```
python convertJunctions.py --samtools samtools –f aln.sam –o
myaln -m 34622 -i ref100/ -s -b -g GRCh37-lite.fa
```

5. Use samtools to create bam file (running convertJunctions with -b and -g does this step automatically):

   `samtools faidx hg19.fa` [index reference genome if not already done so]

   `samtools view –Sb -T hg19.fa myaln.repos.sam > myaln.repos.bam`

   `samtools sort myaln.repos.bam myaln.repos.sorted`

   `samtools index myaln.sorted.repos.bam` [not needed for SNP calling but for IGV]

B) CreateJunctions configuration file example

```
Junction.cfg

[GENOME_REF]
species=homo_sapiens
genome=hg19a
genome_fasta=GRCh37-lite.fa
transcript_file=ens61Genes.txt

[RUN_TIME]
out_dir=./results
seq_length=100

[HEADER_ANNOTATION]
SQ_header_AS=NCBI-Build-37
SQ_header_UR=GRCh37-lite.fa
SQ_header_SP=Homo sapiens
```

This configuration file is required by createJunctions to setup the reference that the RNA-seq reads are aligned to. Under GENOME_REF, information on the species and genome name is required along with the name and path of the genome reference to be used and the transcriptome annotation file. Under RUN_TIME, list the output directory that will hold the reference and indicate the size of the reads that will be used. Header information is listed under HEADER_ANNOTATION.

C) Formatted transcriptome annotation example

A tab delimited transcript text file (referred to above as transcript_file) is required to create the junction reference. Its format currently must be one line per exon for all transcripts in the form of Transcript ID, Chromosome Name (with or without "chr"), Exon Start Coordinate, Exon End Coordinate, Exon Rank in Transcript, Strand) as per below. This can easily be created at http://www.ensembl.org/biomart/ by selecting "Ensembl Genes", the genome dataset of interest, "Attributes" and then "Structures" . Then choose in this order: Ensembl Transcript ID, Chromosome Name, Exon Chr Start (bp), Exon Chr End (bp), Exon Rank in Transcript, Strand. Export all results to a TSV (tab separated values) file. When read, the first header line will be ignored.

Ensembl Transcript ID	Chromosome Name	Exon Chr Start (bp)	Exon Chr End (bp)	Exon Rank in Transcript	Strand
ENST00000416909	13	96185952	96186164	1	-1
ENST00000416909	13	96180746	96180841	2	-1
ENST00000416909	13	96149313	96149379	3	-1
ENST00000416909	13	96131698	96132216	4	-1
ENST00000439928	13	24553949	24554063	1	1
ENST00000439928	13	24557866	24558072	2	1
ENST00000439928	13	24591806	24591840	3	1
ENST00000439928	13	24607652	24607908	4	1
ENST00000439928	13	24608060	24609166	5	1

D) ConvertJunctions input parameters

```
Input: SAM or BAM file of paired end tag (PET) reads aligned to Jaguar
reference/junctions with BWA (sampe -s) and sorted by name
Output: SAM file with genomic coordinates with appropriate reads split by exon

Usage: convertJunctions.py [options]

Options:
  -h, --help            show this help message and exit
  -f file               PET sorted .sam or .bam (f)ile. REQUIRED
  -o o                  (O)utput stub for repositioned file incl. directory,
                        otherwise output in cwd; i.e., [Output tag].repos.sam.
                        REQUIRED
  -i index              (I)ndex folder. REQUIRED
  -m max_separation     (M)ax separation. (M)ax separation
                        recommendations: 34622 is used for hg19a, 23545 is used
                        for hg18 REQUIRED
  -g genome             (G)enome reference fasta file with path. Required if
                        using option -b
  --samtools=SAM_EXEC   SAMTOOLS executable
  -b                    create (B)am after sam conversion
  -c check              chromosome index check, default=1
  -d                    print debugging messages to screen
  -s                    create cigar.txt (CIGAR errors) and .checks (FLAG
                        errors) files
  -x chromosome         filter by comma delimited list of chromosomes, default
                        all chromosomes listed in genome length file
  -l l                  Number (L)ines/reads to reposition from beginning of -f
                        file
  -r r                  pull out read(s) and create reads.sam
  -n                    NO to repositioning, assumes a repositioned file
                        already exists
  -e                    Create bed file of exon-exon junction covered read
                        Coordinates
  --do_not_check_mito_naming
                         if not specified, mitochondrial chromosome M (chrM)
                         will be renamed MT (chrMT)
  --do_not_drop_chr_prefix
                         If not specified, the 'c-h-r' prefix of any chromosome
                         name will be dropped (chr8 becomes 8)
```

E) Example Output after Repositioning

```
JAGuaR Summary
-----------------------------------------------------------------------------
Start:  Wed Mar  6 12:57:48 2014
Finish: Wed Mar  6 18:55:50 2014
File analyzed: testrun.bam
Output file with repositioned reads: testrun.repos.sam
Reference Index Folder: /ref/
Repositioning time: 357.450638203 minutes
Max Separation: 34622

Total number of reads: 246197358
Total number of reads aligned to junctions: 38736275

Junction reads already unmapped in first alignment: 2072703
Junction reads set unmapped after repositioning: 3304
Junction reads mapped and repositioned with large gaps: 36432882
Junction reads repositioned without large gaps: 227940
Number of times one read covers 1, 2, 3 or more junctions:
    1 33254992
    2 3124861
    3 52116
    4 913
Total number of times an exon-exon junction is covered: 39664714
Total number of exon-exon junctions covered by at least one read: 209518
--
```

F) Definition of TLEN implemented in JAGuaR

This program conforms to the Picard spec for TLEN (insert size).
TLEN = distance between 5' ends
TLEN = my mate's start - my start
If a read is +, its start is its POS. If a read is - (ie. it has the 16 bit), its start is POS + the genomic distance the read covers.

For example, if both reads in a pair are -, the TLEN will be calculated as the distance between the rightmost base of read1 and the rightmost base of read2.

If the reads are a normal forward-reverse proper pair, the TLEN will be calculated as the distance between the leftmost base of read1 and the rightmost base of read2

JAGuaR

G) Example Set

A small sample set is available for download and testing:
http://www.bcgsc.ca/downloads/JAGuaR/example

PET fastqs: seq1.fastq.gz, seq2.fastq.gz
Configuration file: junction.cfg
Transcript annotation file: GRCh37p13.txt
Genome reference file: GRCh37-lite.fa.gz
Final bam after JAGuaR run: bwa_mem_aln.bam

1) Download and unzip fastqs and genome fasta file.
2) Download junction.cfg configuration file and example transcript annotation file.
2) Make sure junction.cfg points to correct path of genome reference and transcript file.
3) Run *createJunctions* and index the resulting ref.fa reference as per step 2 in "Pipeline".
4) Align PET fastqs with BWA (or BWA-MEM) to ref.ca (generated above) as per step 3 in "Pipeline".
5) Run *convertJunctions* as per step 4 in "Pipeline".
6) Resulting file should be similar to bwa_mem_aln.bam (also available for download).

Comments, questions and concerns can be sent to:
ybutterf@bcgsc.ca

May, 2014