Boiler Documentation

Release 1.0.1

Jacob Pritt

June 15, 2016

Contents

Boiler is a SAM compression tool designed for downstream isoform assembly and quantitation. Boiler achieves a significant space improvement over other compression tools by discarding unnecessary fields such as sequence information and quality scores. It also tends to shuffle read pairings within highly-covered exons. However, Boiler will always return exact coverage levels over any region of the genome.

Boiler offers several fast queries for its compressed files. Users can query bundles, coverage or reads over any genome interval. Boiler also offers a 'counts' query that returns the read counts over all exons and junctions in a gtf file.

CHAPTER 1

Installing Boiler

You can download the latest version of Boiler from https://github.com/jpritt/boiler. Boiler requires Python 3 or higher to run. There are no other dependencies. You can also run Boiler with PyPy for faster execution.

1.1 Guide

1.1.1 Tutorial

To begin, download the latest version of Boiler from https://github.com/jpritt/boiler. Add the main directory to your path and make sure you have Python version 3 or higher. You will also need SAMtools, which you can download from samtools.sourceforge.net.

Download the SAM dataset here and move it to your working directory.

Run

```
mkdir compressed
python3 boiler.py compress --frag-len-z-cutoff 0.125 accepted_hits.sam compressed/compressed.bl
```

If all goes well, you should see something like this (exact output may change with future versions):

```
Set fragment length cutoff to z=0.125000 (33165) based on length distribution
0.84 % of pairs are longer than the cutoff
Using fragment length cutoff of 33165
Not splitting mates on different strands
Not splitting discordant
0 cross-bundle reads unmatched
Minimum bundle length: 12
Maximum bundle length: 206957
Average bundle length: 2514
1097 cross-bundle buckets
Compressed size: 29682
Approximately 3979761 / 6972093 = 57.081295% of compressed file is coverage
Finished compressing
```

You should now have a file `compressed/compressed.bl` roughly 4.3 MB in size.

Now let's query all of the bundles that Boiler found in chromosome 2L:

```
python3 boiler.py query --bundles --chrom 2L compressed/compressed.bl bundles.txt
```

You should now have a file `bundles.txt` containing all of the bundles used by Boiler. Type

```
head bundles.txt
```

to see the first few lines of this file:

```
7478        9485
9841        21430
21825       23108
23180       24034
24856       25219
25404       26251
26333       33987
34045       35094
36182       37317
37538       37931
```

To query the coverage in the first bundle, run

```
python3 boiler.py query --coverage --chrom 2L --start 7478 --end 9485 compressed/compressed.bl covera
```

`coverage.txt` should now contain a comma-separated vector containing the coverage at every base in the interval [7478, 9485). Finally, to query the reads in the first bundle, run

```
python3 boiler.py query --reads --chrom 2L --start 7478 --end 9485 compressed/compressed.bl reads.sa
```

`reads.sam` is a SAM file with no header, containing all the aligned reads in the interval [7478, 9485). Type

```
head reads.sam
```

to see the first few reads in this bundle, which should look like this:

```
2L:0        0       2L      7772    50      76M     *       0       0       *       *       NH:i:1
2L:1        0       2L      7795    50      76M     *       0       0       *       *       NH:i:1
2L:2        0       2L      7808    50      76M     *       0       0       *       *       NH:i:1
2L:3        0       2L      7863    50      76M     *       0       0       *       *       NH:i:1
2L:4        0       2L      8073    50      44M112N32M      *       0       0       *       *       X
2L:5        0       2L      8595    50      76M     *       0       0       *       *       NH:i:1
2L:6        0       2L      8781    50      76M     *       0       0       *       *       NH:i:1
2L:7        0       2L      8852    50      76M     *       0       0       *       *       NH:i:1
2L:8        0       2L      8963    50      76M     *       0       0       *       *       NH:i:1
2L:9        0       2L      8969    50      76M     *       0       0       *       *       NH:i:1
```

Finally, let's decompress the compressed file by running

```
python3 boiler.py decompress compressed/compressed.bl expanded.sam
```

The resulting SAM file is unsorted – to sort and convert it to BAM, run

```
samtools view -bS expanded.sam | samtools sort - expanded
```

1.1.2 Reference

Boiler has three modes, each described in more detail below:

1. *compress* – compress a SAM file.
2. *query* – query a compressed file.
3. *decompress* – expand a compressed file, outputting a SAM file.

Boiler is invoked by entering

```
python3 boiler.py <mode> <[args]>
```

To run with PyPy instead of Python, run

```
pypy boiler.py <mode> <[args]>
```

To get help for a given mode, enter

```
python3 boiler.py <mode> -h
```

compress

Boiler requires a SAM file to compress. To convert a BAM file to SAM with SAMtools, run:

```
samtools view -h -o path/to/alignments.bam path/to/alignments.sam
```

To compress a SAM file with Boiler, run the following command:

```
python3 boiler.py compress <[args]> path/to/alignments.sam path/to/compressed.bl
```

The following optional arguments are available:

`-c/--frag-len-cutoff <threshold>`

As a first step in compressing, Boiler groups overlapping reads into 'bundles' using a similar method to Cufflinks (see the *bundles* query below for more details). In paired-end datasets, some mates are mapped millions of bases apart or even on different chromosomes. Boiler stores such pairs as bundle-spanning reads, rather than creating massively long bundles to suit them. If `frag-len-cutoff` is set, pairs longer than this cutoff will not contribute to determining bundles, so they will often be stored as bundle-spanning pairs.

Changing the threshold for `--frag-len-cutoff` will not affect accuracy. Generally, decreasing the threshold will lead to faster compression, decompression, and query times, but also to larger file size, and vice versa. See `--frag-len-z-cutoff` below for an alternative to setting the threshold directly.

`-z/--frag-len-z-cutoff <z-score>`

As an alternative to setting `--frag-len-cutoff` directly, if `--frag-len-z-cutoff` is set Boiler will perform a first pass over the reads to establish the average and standard distribution of all paired read lengths. Any reads with a z-score greater than `--frag-len-z-cutoff` will not contribute to determining bundles, so they will often be stored as bundle-spanning pairs. If neither `--frag-len-cutoff` nor `--frag-len-z-cutoff` is set, Boiler will set `--frag-len-z-cutoff` to `0.125`, which we have found to work well in practice.

`-s/--split-diff-strands`

Sometimes a SAM file contains paired mates that lie on different chromosomes. Boiler preserves these pairs by default; use `--split-diff-strands` to convert them to unpaired reads.

`-d/--split-discordant`

SAM files often contain discordant pairs, i.e. paires where one mate intersects an intron of the other mate. Boiler preserves these pairs by default; use `--split-discordant` to convert them to unpaired reads.

`-p/--preprocess`

This argument should be added for alignments produced by HISAT, which require an additional processing step before compression. For multimapped reads, HISAT outputs a list of `n` left mates and `m` right mates, where any left mate may be be paired with any right mate. In contrast, Cufflinks outputs a pair for each unique combination of possible left and right mates. Boiler requires pairs to be enumerated, as in Cufflinks output.

Alternatively, you can preprocess HISAT alignments yourself by running:

```
python enumeratePairs.py --input alignments.sam --output alignments.processed.sam
samtools sort -bS alignments.processed.sam | samtools sort - alignments.processed
samtools view -h -o alignments.processed.sam alignments.processed.bam
```

Following which you can run Boiler as normal:

```
python3 boiler.py compress <[args]> alignments.processed.sam path/to/compressed.bl
```

`-g/--gtf <path/to/transcripts.gtf>`

Boiler offers the option of using a reference gtf file to guide compression. Boiler adds additional splice sites at every transcript splice site and endpoint in the gtf. This improves the accuracy of read recovery at the cost of a significant size increase.

`-v/--verbose`

Print additional debug information.

query

Boiler currently supports the following queries:

1. *bundles*
2. *coverage*
3. *reads*
4. *counts*

The first 3 queries require a chromsome and optional start and end position. Boiler will return the results over the query over the given interval. If `--start` or `--end` are not specified, the endpoints of the given chromsome will be used. Results will be written to the given `out_file`. To run on of these queries, enter:

```
python3 boiler.py [--bundles | --coverage | --reads] --chrom <c> --start <s> --end <e> path/to/compre
```

The `counts` query takes a gtf file as input, but no chromosome or range. Results will be returned for the entire genome. To run this query, enter:

```
python3 boiler.py --counts --gtf <path/to/transcripts.gtf> path/to/compressed.bl out_file
```

bundles

'Bundles' divide the genome into manageable chunks, roughly corresponding to potential gene boundaries. Boiler calculates bundles in a similar way to Cufflinks; as read are processed in order of position, the end of the current bundle is extended to the end of the current paired-end read. If the next read begins more than 50 bases after the end of the current bundle, a new bundle is creating beginning at the current read.

This query prints the bounds of each bundle that overlaps the given range.

coverage

This query prints a vector containing the total coverage at each base in the given range.

reads

This query outputs a SAM file containing all reads that overlap the given range.

counts

Boiler parses the gtf file and extracts a list of exons and junctions

decompress

To decompress a file with Boiler, run

```
python3 boiler.py decompress <[args]> path/to/compressed.bl expanded.sam
```

The output SAM file is not sorted; to convert to a sorted BAM file, enter

```
samtools view -bS expanded.sam | samtools sort - expanded
```

Then you can then sort the expanded SAM file by running

```
samtools view -h -o expanded.sam expanded.bam
```

The following arguments are available for decompression:

`--force-xs`

Boiler will assign a random XS value to any spliced reads that do not have an XS tag. This is meant for some tools such as Cufflinks which require that all spliced reads have an XS tag.

`-v/--verbose`

Print additional debug information.