

Jellyfish: A fast k-mer counter

G. Marcais

February 17, 2012

Version 1.1.4

Abstract

Jellyfish is a software to count k -mers in DNA sequences.

1 Synopsis

```
jellyfish count [-o prefix] [-m merlength] [-t threads] [-s shashsize] [- -both-strands] fasta [fasta ...]  
jellyfish merge hash1 hash2 ...  
jellyfish dump hash  
jellyfish stats hash  
jellyfish histo [-h high] [-l low] [-i increment] hash  
jellyfish query hash  
jellyfish cite
```

Plus equivalent version for *Quake* mode: *qhisto*, *qdump* and *qmerge*.

2 Description

Jellyfish is a k -mer counter based on a multi-threaded hash table implementation.

2.1 Counting and merging

To count k -mers, use a command like:

```
jellyfish count -m 22 -o output -c 3 -s 10000000 -t 32 input.fasta
```

This will count the the 22-mers in input.fasta with 32 threads. The counter field in the hash uses only 3 bits and the hash has at least 10 million entries.

The output files will be named output_0, output_1, etc. (the prefix is specified with the **-o** switch). If the hash is large enough (has specified by the **-s** switch) to fit all the k -mers, there will be only one output file named output_0. If the hash filled up before all the mers were read, the hash is dumped to disk, zeroed out and reading in mers resumes. Multiple intermediary files will be present on the disks, named output_0, output_1, etc.

To obtain correct results from the other sub-commands (such as histo, stats, etc.), the multiple output files, if any, need to be merged into one with the merge command. For example with the following command:

```
jellyfish merge -o output.jf output\_*
```

Should you get many intermediary output files (say hundreds), the size of the hash table is too small. Rerunning *Jellyfish* with a larger size (option **-s**) is probably faster than merging all the intermediary files.

2.2 Orientation

When the orientation of the sequences in the input fasta file is not known, e.g. in sequencing reads, using **--both-strands (-C)** makes the most sense.

For any k -mer m , its canonical representation is m itself or its reverse-complement, whichever comes first lexicographically. With the option **-C**, only the canonical representation of the mers are stored in the hash and the count value is the number of occurrences of both the mer and its reverse-complement.

2.3 Choosing the hash size

To achieve the best performance, a minimum number of intermediary files should be written to disk. So the parameter **-s** should be chosen to fit as many k -mers as possible (ideally all of them) while still fitting in memory.

We consider two examples: counting mers in sequencing reads and in a finished genome.

First, suppose we count k -mers in short sequencing reads: there are n reads and there is an average of 1 error per reads where each error generates k unique mers. If the genome size is G , the size of the hash (option **-s**) to fit all k -mers at once is estimated to: $(G + k * n)/0.8$. The division by 0.8 compensates for the maximum usage of approximately 80% of the hash table.

On the other hand, when counting k -mers in an assembled sequence of length G , setting **-s** to G is appropriate.

As a matter of convenience, Jellyfish understands ISO suffixes for the size of the hash. Hence 's 10M' stands 10 million entries while 's 50G' stands for 50 billion entries.

The actual memory usage of the hash table can be computed as follow. The actual size of the hash will be rounded up to the next power of 2: $s = 2^l$. The parameter r is such that the maximum reprobe value (**-p**) plus one is less than 2^r . Then the memory usage per entry in the hash is (in bits, not bytes) $2k - l + r + 1$. The total memory usage of the hash table in bytes is: $2^l * (2k - l + r + 1)/8$.

2.4 Choosing the counting field size

To save space, the hash table supports variable length counter, i.e. a k -mer occurring only a few times will use a small counter, a k -mer occurring many times will use multiple entries in the hash. The **-c** specify the length of the small counter. The trade off is as follows: a low value will save space per entry in the hash but can potentially increase the number of entries used, hence maybe requiring a larger hash.

In practice, use a value for **-c** so that most of your k -mers require only 1 entry. For example, to count k -mers in a genome, where most of the sequence is unique, use **-c1** or **-c2**. For sequencing reads, use a value for **-c** large enough to count up to twice the coverage.

3 Subcommands and options

3.1 count

Usage: jellyfish count [options] file:path+

Count k-mers or qmers in fasta or fastq files

Options (default value in (), *required):

-m,-mer-len=uint32 *Length of mer
-s,-size=uint64 *Hash size
-t,-threads=uint32 Number of threads (1)
-o,-output=string Output prefix (mer_counts)
-c,-counter-len=Length in bits Length of counting field (7)
-out-counter-len=Length in bytes Length of counter field in output (4)
-C,-both-strands Count both strand, canonical representation (false)
-p,-reprobes=uint32 Maximum number of reprobes (62)
-r,-raw Write raw database (false)
-q,-quake Quake compatibility mode (false)
-quality-start=uint32 Starting ASCII for quality values (64)
-min-quality=uint32 Minimum quality. A base with lesser quality becomes an N (0)
-L,-lower-count=uint64 Don't output k-mer with count \leq lower-count
-U,-upper-count=uint64 Don't output k-mer with count \geq upper-count
-matrix=Matrix file Hash function binary matrix
-timing=Timing file Print timing information
-stats=Stats file Print stats
-usage Usage
-h,-help This message
-full-help Detailed help
-V,-version Version

3.2 stats

Usage: jellyfish stats [options] db:path

Statistics

Display some statistics about the k-mers in the hash:

Unique: Number of k-mers which occur only once. Distinct: Number of k-mers, not counting multiplicity. Total: Number of k-mers, including multiplicity. Max_count: Maximum number of occurrence of a k-mer.

Options (default value in (), *required):

- L, -lower-count=uint64** Don't consider k-mer with count \leq lower-count
- U, -upper-count=uint64** Don't consider k-mer with count \geq upper-count
- v, -verbose** Verbose (false)
- o, -output=c_string** Output file
- usage** Usage
- h, -help** This message
- full-help** Detailed help
- V, -version** Version

3.3 histo

Usage: jellyfish histo [options] db:path

Create an histogram of k-mer occurrences

Create an histogram with the number of k-mers having a given count. In bucket 'i' are tallied the k-mers which have a count 'c' satisfying ' $\text{low} + i * \text{inc} \leq c < \text{low} + (i+1) * \text{inc}$ '. Buckets in the output are labeled by the low end point ($\text{low} + i * \text{inc}$).

The last bucket in the output behaves as a catchall: it tallies all k-mers with a count greater or equal to the low end point of this bucket.

Options (default value in (), *required):

- l, -low=uint64** Low count value of histogram (1)
- h, -high=uint64** High count value of histogram (10000)
- i, -increment=uint64** Increment value for buckets (1)
- t, -threads=uint32** Number of threads (1)
- f, -full** Full histo. Don't skip count 0. (false)
- o, -output=c_string** Output file
- v, -verbose** Output information (false)
- usage** Usage

- **-help** This message
- **-full-help** Detailed help
- **-V,- -version** Version

3.4 query

Usage: jellyfish query [options] db:path

Query from a compacted database

Query a hash. It reads k-mers from the standard input and write the counts on the standard output.

Options (default value in (), *required):

- **-C,- -both-strands** Both strands (false)
- **-c,- -cary-bit** Value field as the cary bit information (false)
- **-i,- -input=*file*** Input file
- **-o,- -output=*file*** Output file
- **-usage** Usage
- **-h,- -help** This message
- **-V,- -version** Version

3.5 cite

Usage: jellyfish cite [options]

How to cite Jellyfish's paper

Citation of paper

Options (default value in (), *required):

- **-b,- -bibtex** Bibtex format (false)
- **-o,- -output=*c_string*** Output file
- **-usage** Usage
- **-h,- -help** This message
- **-V,- -version** Version

4 Version

Version: 1.1.4 of February 17, 2012

5 Bugs

- *jellyfish merge* has not been parallelized and is relatively slow.
- The hash table does not grow in memory automatically and *jellyfish merge* is not called automatically on the intermediary files (if any).

6 Copyright & License

Copyright © 2010, Guillaume Marcais `guillaume@marcais.net` and Carl Kingsford `carlk@umiacs.umd.edu`.

License This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program.

If not, see <http://www.gnu.org/licenses/>.

7 Authors

Guillaume Marcais
University of Maryland
`gmarcais@umd.edu`

Carl Kingsford
University of Maryland
`carlk@umiacs.umd.edu`